数值分析

第七章：非线性方程（组）数值解

张亚楠[1]

苏州大学数学科学学院

April 30, 2020

[1]Email: ynzhang@suda.edu.cn

Contents

必要性

高次代数方程和超越方程一般没有求根公式

已知 $f(x) \in C_{[a,b]}$, $f(a)f(b) < 0$, 如何求解

$$f(x) = 0 \tag{1}$$

- 二分法
- 不动点迭代(理论)
- Newton法及其变形
- Newton法推广：非线性方程组
- Inexact Newton 法

如果存在实数x^* 满足$f(x^*) = 0$, 则称 x^* 是方程的根，或者说是函数f(x)的零点. 如果f(x)可以分解成

$$f(x) = (x - x^*)^m g(x)$$

且 $g(x^*) \neq 0$，则称x^* 是f(x)的m重零点，或方程的m重根.

求解之前，先确定解的存在区间$[a,b]$

Example 1

求方程的根的存在区间

$$f(x)=100*(3^x-1)-1200*x=0$$

$f(1)$	$f(2)$	$f(3)$	$f(4)$	$f(5)$	$f(6)$	$f(7)$	$f(8)$	$f(9)$	$f(10)$
-1000	-1600	-1000	3200	18200	65600	210200	646400	1957400	5892800

Example 2

求方程 $f(x) = x^3 - 11.1x^2 + 38.8x - 41.77 = 0$在[1,3]区间的根

精确解：2.096315198390627

计算得到 $f(1) < 0$, $f(3) > 0$, 根据零点定理，区间[1,3]必然存在一个根. 称之为根的存在区间$[a_0, b_0]$ 区间长度为2

接下来计算函数f在区间中点的函数值，得到$f(2) < 0$, 则新的根存在区间为 [2,3], 记为 $[a_1, b_1]$ 区间长度为1

继续 计算函数f在区间中点的函数值，得到$f(2.5) > 0$, 则新的根存在区间为 [2,2.5], 记为 $[a_2, b_2]$ 区间长度为1/2

继续上述过程可以得到由根的存在区间生成的闭区间套

$$[a_0, b_0] \supset [a_1, b_1] \supset [a_2, b_2] \cdots \supset [a_n, b_n] \cdots$$

观察发现每得到一个新的根存在区间，区间长度缩小一半。 思考：随着 n 越来越大，区间长度会怎样?

求解之前，先确定解的存在区间$[a,b]$, 李庆扬教材p213 例1

$f(a)*f(b)<0$, 不妨假设$f(a)<0, f(b)>0$

1. $x_0=\frac{a+b}{2}$, and comput $f(x_0)$

2. if $f(x_0)=0$

   output $x_0=x^*$

elseif $f(x_0)>0$

   set $\cancel{a=a};b=x_0$

else ($f(x_0)<0$)

   set $a=x_0;\cancel{b=b}$

endif

3. return to 1

Example 3

求方程 $f(x)=x^3-11.1x^2+38.8x-41.77=0$在[1,3]区间的根

精确解：2.096315198390627

对分得到的结果如下：

$$\begin{array}{cccccc} 2.0000 & 2.5000 & 2.2500 & 2.1250 & 2.0625 & 2.0938 \\ 2.1094 & 2.1016 & 2.0977 & 2.0957 & 2.0967 & 2.0962 \end{array}$$

1. 对分方法一定收敛：随对分次数增加，误差越来越小
2. 二分法收敛较慢：对比后文Newton法（只需几步）

Tips : Matlab 演示一下二分法算例！runbisec.m

Contents

将 f(x) = 0 写成等价形式

$$x = \varphi(x)$$

若存在 x^*满足上式，则称之为$\varphi(x)$ 的一个不动点。

求 f(x)的零点等价于 求 $\varphi(x)$ 的不动点。

由等价形式构造迭代格式

$$x_{k+1} = \varphi(x_k), \quad k = 0, 1, 2, ...$$

$\varphi(x)$称为迭代函数。

给定初值x_0

$$x_{k+1} = \varphi(x_k), \quad k = 0, 1, 2, ...$$

反复作用上式，得到数列

$$x_0, x_1, x_2, ..., x_k, ...$$

如果对任何$x_0 \in [a, b]$，上述数列收敛，即

$$\lim_{k\to\infty} x_k = x^*$$

则称迭代格式收敛，且称

$$x^* = \varphi(x^*)$$

为$\varphi(x)$ 的不动点，上述方法称为不动点迭代法。

Example 4

改写上例

$$f(x) = 100 * (3^x - 1) - 1200 * x = 0$$

为

$$3^x = 12 * x + 1 \to x = ln(12 * x + 1)/ln3$$

构造简单迭代格式，并观察收敛性

Matlab演示！simp_itr1.m

问题：迭代格式何时收敛？ 回忆线性方程组迭代法的收敛条件！

Theorem 5

设 $\varphi(x) \in C[a,b]$满足以下条件

(1) 对任意$x \in [a,b]$, $\varphi(x) \in [a,b]$

(2) 存在正常数$L < 1$, 使得对任意$x, y \in [a,b]$，都有

$$|\varphi(x) - \varphi(y)| \leq L|x - y|$$

则 在 上存在唯一不动点x^*。

Hint：(存在性证明) 构造辅助函数

$$h(x) = \varphi(x) - x$$

$h(x)$ 连续，有 $h(a) \geq 0$, $h(b) \leq 0$ 介值定理即得结论.

设x^*是唯一的不动点，则

$$x^* = \varphi(x^*)$$

迭代格式

$$x_{k+1} = \varphi(x_k)$$

记第k步的误差

$$e_k = x_k - x^*$$

则

$$|x_k - x^*| = |\varphi(x_{k-1}) - \varphi(x^*)| \leq L|x_{k-1} - x^*| \leq ... \leq L^k|x_0 - x^*|$$

进而 误差满足

$$|e_k| \leq L^k * |e_0| \to 0, \quad as \quad (L < 1)\ \&\ k \to \infty$$

误差估计

1. 先验估计:

$$|x_k - x^*| \le \frac{L^k}{1-L}|x_1 - x_0|$$

2. 后验估计:

$$|x_k - x^*| \le \frac{1}{1-L}|x_{k+1} - x_k|$$

建议:

同学自行推导，与线性方程组迭代法误差估计类似，这里不涉及向量，更简单。

Definition 6

若存在 $\varphi(x)$ 的不动点x^*的某个领域$U(x^*,\delta)$，对任意的$x \in U$，迭代法产生的序列$\{x_k\}$均收敛，则称迭代法局部收敛。

可以证明：

Theorem 7

若$\varphi^{'}(x)$ 在x^* 的某个邻域连续，且$|\varphi^{'}(x)| < 1$，则迭代法局部收敛。

想一想：

对$f(x) = 0$ 可以构造不同的迭代格式，有的收敛有的不收敛，收敛速度也不同。总不能漫无目的的尝试吧，有没有一个统一的、有效方法？

Newton method ！！！

Definition 8

若迭代格式$x_{k+1}=\varphi(x_k)$收敛于x^*，记第k步迭代误差

$$e_k = x^* - x_k$$

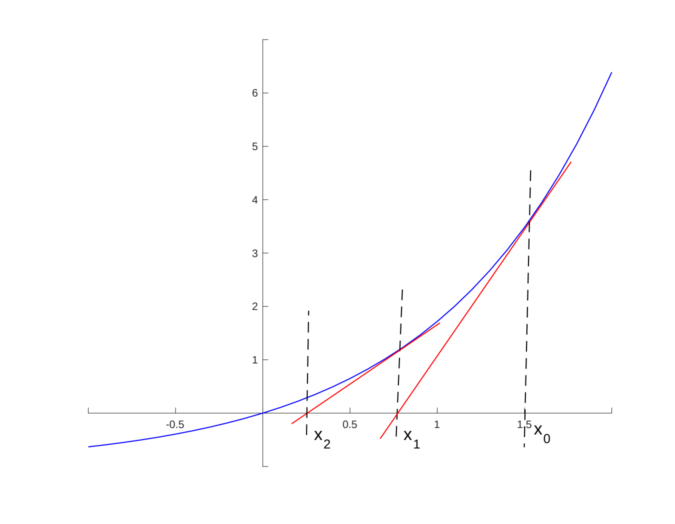
若迭代误差满足

$$\lim_{k\to\infty}\frac{e_{k+1}}{e_k^p} = C \geq 0$$

则称迭代格式p阶收敛。当p = 1时称为线性收敛，p = 2时称为平方收敛.

Contents

Newton法的几何解释



给定x_k,如何由切线方法更新近似解？ 切线方程：

$$Y - f(x_k) = f^{'}(x_k)(X - x_k)$$

计算切线与x轴交点即$Y = 0$, 得到

$$0 - f(x_k) = f^{'}(x_k)(X - x_k), \rightarrow X = x_k - \frac{f(x_k)}{f^{'}(x_k)}$$

作为下一步更新值

$$\begin{cases} x_{k+1} = x_k - \frac{f(x_k)}{f^{'}(x_k)}, \quad f^{'}(x_k) \neq 0 \\ x_0 = \text{ initial guess} \end{cases}$$

由

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

得到

$$\left(x_{k+1} - x_k\right) * f^{'}(x_k) = -f(x_k)$$

也可将迭代格式改写为如下形式

Newton法标准形式

$$\begin{cases} f^{'}(x_k) * \boxed{s} = -f(x_k) \\ x_{k+1} = x_k + \boxed{s} \end{cases}$$

想一想：每次迭代的主要运算量在哪里？

Example 9

构造

$$f(x) = 100 * (3^x - 1) - 1200 * x = 3^x - 1 - 12x = 0$$

的牛顿迭代格式。

记

$$df = f^{'}(x) = 3^x ln3 - 12,$$

演示Matlab计算效果，文件名 NewtonItr.m

说明

1. 同一个方程可以有不同的迭代 格式；
2. Newton法收敛速度较快. WHY?
3. Newton法局部收敛，因此对初值要求较高；可以结合二分法来用

课后作业：Newton法对应的迭代函数φ是什么？导数多大？

想一想：

如果Newton迭代法中$f^{'}(x)$表达式较为复杂，如何构造简单有效的迭代法？

<u>差商替代导数</u> 利用

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

由Newton公式

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

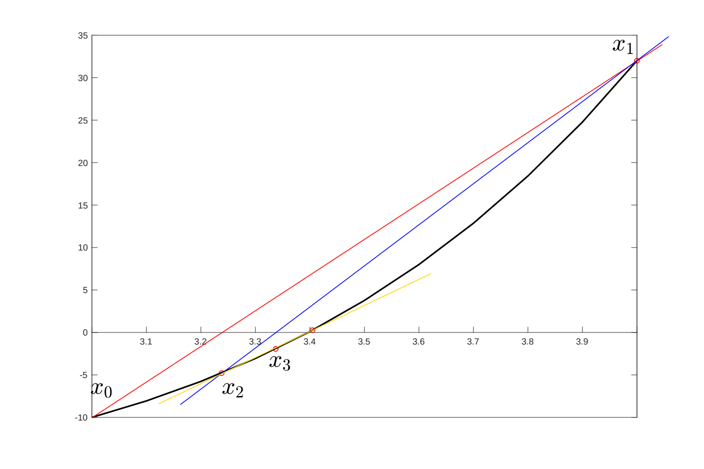
可得

$$x_{k+1} = x_k - \frac{(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})} f(x_k)$$

$$\boxed{\begin{cases} \text{Solve} & \dfrac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} * s = -f(x_k) \\ \text{Update} & x_{k+1} = x_k + s \end{cases}}$$

上述算法避开了求导数。 除了差商近似导数，有无其它对算法可靠性的简单解释？

割线法的几何解释



弦截法超线性收敛!! 演示 Matlab举例计算前例，效果也不错！newtonitr.m

1. 割线法超线性收敛: 收敛阶1.618, 好神奇！！
2. Newton法有其他变形：Newton下山、简单Newton
3. 抛物线法($Müller$): 收敛阶1.84，优点是可求复根

Tips :

1. 良态问题： 简单！Newton法或者割线法可以满足精度需求。
2. 病态问题：难！！！何为病态?

求多项式零点也可看作是数集（系数）到数集（零点）的映射。输入到输出！ 病态就是输入数据扰动反应到输出数据上，过分放大！！

求多项式零点也可看作是数集（系数）到数集（零点）的映射。假设输入有误差，考查输出误差如何变化?

Example 10

假设多项式的某个系数有扰动

$$
\begin{aligned}
p(x) &= (x-1)(x-2)(x-3)(x-4)(x-5)(x-6)(x-7) \\
&= x^7 - 28x^6 + 322x^5 - 1960x^4 + 6769x^3 - 13132x^2 + 13068x - 5040
\end{aligned}
$$

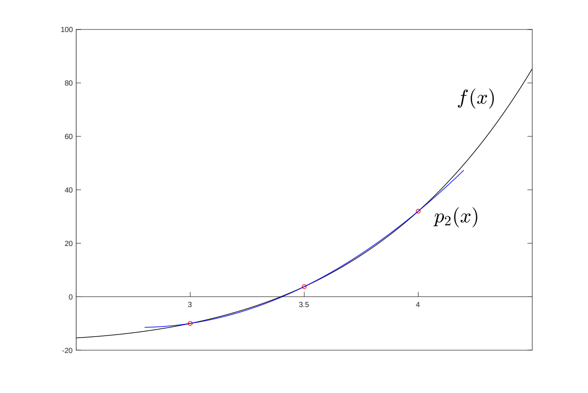
假设第二项系数有近似万分之一的相对扰动$c_2 = -28.0028$, 则求出根为

$$
\begin{aligned}
&7.2994 + 0.0000i \\
&5.4476 + 0.6748i \\
&5.4476 - 0.6748i \\
&3.7615 + 0.0000i \\
&3.0483 + 0.0000i \\
&1.9985 + 0.0000i \\
&1.0000 + 0.0000i
\end{aligned}
$$

1. Newton法和割线法都是采用“以直代曲”思想
2. Muller方法的核心思想：以抛物线近似曲线

$M\ddot{u}ller$ 方法

1. Newton法和割线法都是采用“以直代曲”思想
2. Muller方法的核心思想：<u>以抛物线近似曲线</u>



1. Newton法和割线法都是采用“以直代曲”思想
2. Muller方法的核心思想：<u>以抛物线近似曲线</u>

假设已知三个近似零点x_0, x_1, x_2, 如何利用抛物线近似曲线$f(x)$, 并更新零点。

1. 构造二次插值多项式(Vandermonde矩阵)

$$\begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} \rightarrow p_2(x) = c_0 + c_1 x + c_2 x^2$$

2. 计算上式零点(有求根公式)，二次方程有两个根 z_1, z_2 还有可能出现复数
3. 挑一个合适的z作为x_3，迭代更新

Matlab：演示！！mullerMeth.m

Tips： 如果明确知道没有复根，只是为了求出实根，抛物线法似无必要。割线法很好！！

任意的多项式求零点可以转化为首一(monic) 的多项式

$$x^n + c_1 x^{n-1} + c_2 x^{n-2} + ...c_{n-1}x + c_n = 0.$$

Example 11

上式左端多项式是矩阵

$$C = \begin{bmatrix} -c_1 & -c_2 & \cdots & -c_n \\ 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 \end{bmatrix}$$

的特征多项式。

~~对行列式$det(\lambda I - C)$按行展开，检验即可。~~

若求出矩阵 C 的特征值，即可得到上述多项式的零点。

想一想：

是否觉得《线性代数》和《数值分析》在相互推诿、踢皮球？

1. 如何求特征值？线性代数说：特征多项式求根！
2. 多项式如何求根？数值分析说：求对应矩阵的特征值！

解决方案：数值分析有其他手段求矩阵特征值！！！注意体会“理论数学”和“计算数学”的异同！！！

小结

1. 简单问题，什么方法都好使
2. 病态问题，相对困难，也勉强可行：实根Newton法、割线法；复根抛物线法
3. 多项式求根，可以利用后文的特征值计算手段，效果更优！

考虑二阶方程组 (n阶方程组可类似分析)

$$\begin{cases} f_1(x_1, x_2) = 0 \\ f_2(x_1, x_2) = 0 \end{cases}$$

上式可以写成向量形式:

$$\boldsymbol{F}(\boldsymbol{x}) = \vec{\mathbf{0}}$$

为了书写简单，不再用黑体字，简记为

$$F(x) = 0$$

Notice:

非线性方程组比单个非线性方程或者线性方程组要复杂的多； 同时也在实际问题中经常出现，有广泛的应用。类似可以考虑n个未知量情形, n 越大问题越复杂！！

Newton法

回顾单变量的Newton法：

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \Leftrightarrow x_{k+1} = x_k - [f'(x_k)]^{-1} f(x_k)$$

形式上方程组也有类似结果

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \left[F'(\boldsymbol{x}_k)\right]^{-1} F(\boldsymbol{x}_k)$$

定义Jacobi矩阵

$$J(\boldsymbol{x}) = F'(\boldsymbol{x}) = \begin{bmatrix} \dfrac{\partial f_1(\boldsymbol{x})}{\partial x_1} & \dfrac{\partial f_1(\boldsymbol{x})}{\partial x_2} \\ \dfrac{\partial f_2(\boldsymbol{x})}{\partial x_1} & \dfrac{\partial f_2(\boldsymbol{x})}{\partial x_2} \end{bmatrix}$$

则非线性方程组的Newton迭代格式如下

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \left[F'(\boldsymbol{x}_k)\right]^{-1} F(\boldsymbol{x}_k)$$

记导数矩阵

$$F^{'}(x)=\begin{bmatrix}\dfrac{\partial f_1(\boldsymbol{x})}{\partial x_1} & \dfrac{\partial f_1(\boldsymbol{x})}{\partial x_2}\\ \\ \dfrac{\partial f_2(\boldsymbol{x})}{\partial x_1} & \dfrac{\partial f_2(\boldsymbol{x})}{\partial x_2}\end{bmatrix}$$

Newton迭代格式

$$\boldsymbol{x}_{k+1}=\boldsymbol{x}_k-\left[F^{'}(\boldsymbol{x}_k)\right]^{-1}F(\boldsymbol{x}_k)$$

转化为

$$F^{'}(\boldsymbol{x}_k)*\boldsymbol{x}_{k+1}=F^{'}(\boldsymbol{x}_k)*\boldsymbol{x}_k-F(\boldsymbol{x}_k)$$

进一步

$$\boxed{\begin{cases}\color{red}{F^{'}(\boldsymbol{x}_k)*\vec{s}}=-F(\boldsymbol{x}_k),\\ \boldsymbol{x}_{k+1}=\boldsymbol{x}_k+\vec{s}\end{cases}}$$

红色部分是求解线性方程组$Ax=b$. 想一想：上述过程，那部分计算量最大？

Example 12

(教材例子) 计算$\boldsymbol{F(x)=0}$, 取$x_0=[0,0]^{\mathrm{T}}$

$$F(x)=\begin{bmatrix} x_1^2-10x_1+x_2^2+8 \\ x_1x_2^2+x_1-10x_2+8 \end{bmatrix}, \quad F^{'}(x)=\begin{bmatrix} 2x_1-10 & 2x_2 \\ x_2^2+1 & 2x_1x_2-10 \end{bmatrix}$$

Matlab： newtonitr.m

$$\boxed{\begin{cases} F^{'}(\boldsymbol{x}_k)*\vec{s}=-F(\boldsymbol{x}_k), \\ \boldsymbol{x}_{k+1}=\boldsymbol{x}_k+\vec{s} \end{cases}}$$

$$\boxed{\begin{cases} F'(\boldsymbol{x}_k) * \vec{s} = -F(\boldsymbol{x}_k), \\ \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \vec{s} \end{cases}}$$

想一想：

如果未知量100个，程序咋写？先手算一万个偏导函数？再输到电脑里？

1. 实际问题都有一定的规律，算子一般是局部的，不会这么夸张，但计算Jabocian 总归很麻烦！！！
2. Newton法的每一步要求解线性方程组，迭代的每一步系数矩阵都在变化!! 当未知量个数很大时，求解很困难！！
   - 直接法：预先的LU分解没法用、因为每步迭代都是对应新的系数矩阵
   - 迭代法：同样没有统一的预处理，每步的预处理也随着系数矩阵的不同而改变

总之、未知量个数多了，Newton法实现较为困难。如果特别多（n > 1百万？），可以放弃精确Newton法。

1: SET initial guess $\boldsymbol{x}_0$

2: FOR $k = 1, 2, 3, ...$ DO

Find a vector $\boxed{s}$ satisfied

$$F^{'}(\boldsymbol{x}_k) * s = -F(\boldsymbol{x}_k) + r_k, \quad \frac{\|r_k\|}{\|F(x_k)\|} < \cancel{\eta = 0.1 < 1}$$

3: SET $x_{k+1} = x_k + s$

核心思想：避免计算$F^{'}(x_k)$，寻找替代品，例如有限差分

$$F^{'}(x_k)s = \frac{F(x_k + \sigma s) - F(x_k)}{\sigma}$$

注意到第二步（红色部分），本质上是求解线性方程组 $As = b$，而许多迭代法如Krylov子空间法， 求解$As = b$的过程，是反复在处理残量$r_m = b - As_m$. 假如可以找到 As_m 的有效近似， 则可得残量r_m的近似。进而求出更新向量s的有效近似值。Good！

Contents

1: SET initial guess $\boldsymbol{x}_0$

2: FOR $k = 1, 2, 3, \ldots$ DO

Find a vector $\boxed{s}$ satisfied

$$As = F^{'}(\boldsymbol{x}_k) * s = -F(\boldsymbol{x}_k) + r_k, \quad \rightarrow \boxed{\mathcal{L}s = -F(\boldsymbol{x}_k)}$$

3: SET $x_{k+1} = x_k + s$

取定差分步长(例如$\sigma = 1e-5$)

$$F^{'}(x_k)s \approx \boxed{\mathcal{L}s := \frac{F(x_k + \sigma s) - F(x_k)}{\sigma}}$$

结合FOM(Arnoldi full orthogonal method) 方法求解近似方程组

$$As = -F(\boldsymbol{x}_k) \rightarrow \mathcal{L}s = -F(\boldsymbol{x}_k)$$

可得 s.

差分近似

$$F^{'}(x_k)s \approx \boxed{\mathcal{L}s := \frac{F(x_k+\sigma s)-F(x_k)}{\sigma}}$$

Example 13

取定$x_k = [0.8; 0.8]$, $\sigma = 10^{-4}$, 检验差法近似对向量$s = [0.1; 0.1]$ 的近似效果

$$F(x) = \begin{bmatrix} x_1^2 - 10x_1 + x_2^2 + 8 \\ x_1x_2^2 + x_1 - 10x_2 + 8 \end{bmatrix}, \quad F^{'}(x) = \begin{bmatrix} 2x_1 - 10 & 2x_2 \\ x_2^2 + 1 & 2x_1x_2 - 10 \end{bmatrix}$$

$$F^{'}(x_k)s = \begin{bmatrix} -8.4000 & 1.6000 \\ 1.6400 & -8.7200 \end{bmatrix} * \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} = \begin{bmatrix} -0.680000000000000 \\ -0.708000000000000 \end{bmatrix}$$

$$\mathcal{L}s = \begin{bmatrix} -0.679998000006066 \\ -0.707997599986854 \end{bmatrix}, \qquad \|F^{'}(x_k)s - \mathcal{L}s\| = 3.1241 \times 10^{-6}$$

Matlab演示：中心差分效率更高！Diff_jacobi.m

取定差分近似

$$F^{'}(x_k)s \approx \boxed{\mathcal{L}s := \frac{F(x_k + \sigma s) - F(x_k)}{\sigma}}$$

算法

$$\boxed{\begin{cases} \mathcal{L}\vec{s} = -F(\boldsymbol{x}_k), \\ \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \vec{s} \end{cases}}$$

注意到矩阵乘向量As是一个$\boldsymbol{R}^n \to \boldsymbol{R}^n$ 的映射， 简言之，输入一个向量$\boxed{s}$, 返回一个向量$\boxed{As}$. 而差分近似$\mathcal{L}s$同样具有这一功能。

一句话：能算矩阵乘向量Ax_0, 就可以解$Ax = b$. 同样， $\mathcal{L}s = b$ 也可求解！！！HOW?

$\boxed{\text{Krylov子空间方法！！！}}$

Contents

Arnoldi FOM （Full Orthogonal Method）求解 $Ax = b$

Krylov子空间

$$\mathcal{K}_m(A, v) = span\{v, Av, A^2v, ..., A^{m-1}v\}$$

给定初值猜测x_0, 进而可得残量$r_0 = b - Ax_0$, 我们尝试在Krylov子空间

$$\mathcal{K}_m\Big(A, r_0\Big) = span\{r_0, Ar_0, A^2r_0, ..., A^{m-1}r_0\}$$

中寻找近似解（CG方法已知道上述空间有好的近似解！！！）~~并提出要求~~

$$x_m = x_0 + z_k \underline{\in \mathcal{K}_m}, \quad \& \quad r_m = b - Ax_m \perp \mathcal{K}_m$$

可行性OK

目标搜寻空间

$$\mathcal{K}_m\left(A, r_0\right) = span\{r_0, Ar_0, A^2 r_0, ..., A^{m-1} r_0\}$$

选择0初值则$b = r_0$.

几点说明

1. $\boldsymbol{R}^n$的子空间$\mathcal{K}_m\left(A, b\right)$ 容易得到！
2. 第4章：（离散的最佳平方逼近）的知识说明：子空间里面应该有“最优解”
3. HOW？如果知道了这个子空间的一组正交基呢？

假设已知$\mathcal{K}_m\left(A, b\right)$的一组正交基$v_j$, 将其组装为一个 $(n \times m)$ 的瘦长型矩阵 V

$$V = [v_0, v_1, ..., v_{m-1}]$$

在$\mathcal{K}_m\left(A, b\right) = C(V)$, 空间中怎么找$x$ 的最佳近似?

$$\forall x \in C(V), \quad \to \boxed{x = V * y} = \sum_{j=0}^{m-1} y_j * \vec{v}_j, \quad y \in \boldsymbol{R}^m$$

进而

$$Ax = b \to A * V * y = b$$

新的系数矩阵$A * V$ 行列大小? 是否是瘦长型的? 如果是，怎么求解?

$$\boxed{\begin{cases} \left(V^{\mathrm{T}} AV\right) * y = b \\ x = V * y \end{cases}}$$

万事具备，只欠“正交基”！！！ 怎样找 V ？ 问 Arnoldi ！！

给定向量r_0, 矩阵A, 先给$\mathcal{K}_m(A,\ r_0)$空间找一组标准正交基

1: SET $v_1 = \dfrac{r_0}{\|v_0\|}$

2: FOR $k = 1 : m$

$$w_k = \boxed{A * v_k}$$

将w_k 依次对 $\{v_j\}$进行正交投影

$$w_k = w_k - \sum_{j=1}^{k} \Big(w_k, v_j\Big) v_j \quad h_{jk} = \Big(w_k, v_j\Big)$$

3: 单位化 $v_{k+1} = \dfrac{w_k}{\|w_k\|}, \quad h_{k+1,k} = \|w_k\|$

上述产生的v_j, 是子空间$\mathcal{K}_m(A, r_0)$的一组标准正交基

$$span\{v_1, v_2, ..., v_m\} = span\{r_0, Ar_0, ..., A^{m-1} r_0\}$$

上述过程的计算量主要来自于m次矩阵乘向量，如果A是稀疏矩阵，这计算量$\boldsymbol{O}(m^2 n)$

将算法产生的单位正交向量v和内积h 组装为矩阵

$$V_m = \Big[v_1, v_2, ..., v_m\Big]_{n\times m}, \quad H_{m\times m} = h_{jk}$$

则成立

$$\begin{aligned} AV_m &= V_m H_m + w_m e_m^{\mathrm{T}} && (2)\\ &= V_{m+1}\bar{H}_m && (3)\\ V_m^{\mathrm{T}} A V_m &= H_m && (4) \end{aligned}$$

Notice:

1. 给定向量r_0, 和矩阵A, 即可产生V, H 并满足上述等式
2. 算法过程中只用到$A * v_j$
3. 上述结论的证明(可参考Yousef Saad著作Iteration methods for sparse linear systems)

Matlab : myArnoldi1.m （与QR分解相似，但不同！！）

给定初值x_0, 可以取$x_0 = \mathbf{0}$, 则$r_0 = b$, 由Arnoldi算法即可得到V_m，H_m且满足：

$$\boxed{V_m^{\mathrm{T}} A V_m = H_m}$$

计算 $Ax = b$（~~粗略推导，不是严格数学过程！！~~）

$$V_m^{\mathrm{T}} A\underline{x} = V_m^{\mathrm{T}} * b \Rightarrow V_m^{\mathrm{T}} A \underline{V_m * y} = V_m^{\mathrm{T}} * b$$

Arnoldi算法得到 V_m，H_m, 则

$$\boxed{\begin{cases} H_m y = V_m^{\mathrm{T}} * b \\ x = V_m * y \end{cases}}$$

上述过程是否成立，取决于x 是否在 V_m的列空间，也即是$\mathcal{K}_m(A, r_0)$

Tips : $m \approx n$ 时，效果很好！ 如果非线性方程组的未知量个数非常大（成千上万），也可采用Krylov子空间方法如GMRES算法近似求解。 如果未知量只有几百个 或者更少，Arnoldi FOM（Full Orthogonal Method）完全可以接受。

演示：Matlab： myFOM1.m 举例

回到待解决问题

1: Give $\boldsymbol{x}_0$

2: FOR $k = 1, 2, 3, \ldots$

find a vector $\boxed{s}$ satisfied

$$As \approx \mathcal{L}s = -F(\boldsymbol{x}_k)$$

3: SET $x_{k+1} = x_k + s$

Arnoldi FOM 方法可以求解$As = b$, 算法中只需要反复计算$A * v$并做投影即可。 这里要解决的问题

$$\boxed{\mathcal{L}s = b}$$

类似地，只要反复计算 $\mathcal{L}v$并投影即可。

给定向量r_0, ~~矩阵A,~~ 算子$\mathcal{L}$

1: SET $v_1 = \dfrac{r_0}{\|v_0\|}$

2: FOR $k = 1 : m$

$$\cancel{w_k = \boxed{A * v_k}} \rightarrow w_k = \mathcal{L} v_k$$

将w_k 依次对 $\{v_j\}$进行正交投影

$$w_k = w_k - \sum_{j=1}^{k} \Big(w_k, v_j\Big) v_j \quad h_{jk} = \Big(w_k, v_j\Big)$$

3: 单位化 $v_{k+1} = \dfrac{w_k}{\|w_k\|}, \quad h_{k+1,k} = \|w_k\|$

上述产生的v_j, 是子空间$\mathcal{K}_m(\mathcal{L}, r_0)$的一组标准正交基

$$span\{v_1, v_2, ..., v_m\} = span\{r_0, \mathcal{L}r_0, ..., \mathcal{L}^{m-1} r_0\}$$

其它与FOM方法求解 $As = b$ 一样！ Matlab 演示：NewtonItrsysm.m chap7Ex3.m

单个方程求根

1. 二分法简单有效，~~精度不高?~~
2. Newton法高效但对初值敏感，有许多变形：如割线法(推荐！！)
3. Muller 方法可以求解复根（若为了求实根，不推荐！）

n维方程组情形：Newton法直观, 高效！！

1. 若 $n = 2, 3, 4$? 手算也推荐！！
2. 若 $n \approx 10$，100, 可以推荐！看个人的接受程度 和 Jacobi 矩阵的计算难度？
3. 若 $n > 10^4$, 推荐 Inexact Newton method ！！！ 算法相对复杂！

Newton法应用非常广泛，例如求函数的驻点（对应物理上某些能量的稳定态和过渡态！） 此时的$n \approx 10^6$, 必须关心算法效率，求解的每一步都要当心。Arnoldi FOM 求解 $\boxed{\mathcal{L}s = b}$ 不再适用; 但是将这一过程改为gmres方法。Inexact Newton method依然有效。

给定能量函数

$$E(U) = \frac{1}{2}\int_{\Omega} \kappa|\nabla U|^2 + f(U)d\Omega$$

对函数U进行差分离散后，U即为n维向量；上述能量函数的“驻点”即满足

$$\boxed{\kappa\Delta_h\vec{U} + f^{'}(\vec{U}) = 0}$$

Δ_h 即是差分矩阵（不同的边界条件略有不同）， 上式即 n维非线性方程组求根问题: n的大小取决于网格剖分数和问题维度， 以单位长度100等分为例，空间 1 到 3D问题对应的未知量个数分别是1百，1万，1百万。 如对精度要求很高，可以考虑利用Inexact-Newton法求解该问题。

对该驻点的计算通常选择 简单有效的 Euler 方法

$$U^{k+1} = U^k + dt * \left(\kappa\Delta_h U^k + f^{'}(U^k)\right)$$

计算到稳定态即可。（可以理解为梯度下降法）
为了说明Newton法 收敛快、Inexact-Newton法的有效性，我们可以对比Euler方法和 “ Finite difference - Arnoldi - FOM - Inexact Newton ” 方法的迭代步数差别。

Matlab code : testnewton1dch.m   Inexactnewton1dch.m