Chapter 7　非线性方程(组)数值解

张亚楠[*]

问题提出：

高次代数方程和超越方程一般没有求根公式

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

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

- 二分法
- 不动点迭代：Newton法及其变形
- 推广到方程组情形

一些记号和名词

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.

1　二分法

求解之前，先确定解的存在区间$[a, b]$, 李庆扬教材p213 例1

例 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

Matlab code: runbisec.m

*ynzhang@suda.edu.cn　苏州大学数学科学学院

Algorithm 1 二分法　$f(a)*f(b)<0$, 不妨假设$f(a)<0, f(b)>0$

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

2: compute $f(x_0)$

 if $f(x_0)=0$

 Get the solution $x_0=x^*$,　and stop.

 elseif $f(x_0)>0$

  set $\cancel{a=a;}$　$b=x_0$

 elseif $f(x_0)>0$

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

 endif

3: return to 1

```
% --- bisection meth
f = @(x) 100*(3.^x - 1) - 1200*x;
% t1 = 3:0.01:4; f1 = f(t1);
% figure(1);plot(t1,f1,'k-'); hold on,
x0 = 3; x1 = 4;  maxI = 30;
for k = 1:maxI
    md = (x0 + x1)/2;
    f0 = f(md);
    if abs(f0) < 1e-9
      break
    elseif f0 > 0
        x1 = md;
    else
        x0 = md;
    end
end
fprintf('\n␣the␣itr␣stop␣at␣%d,␣x␣=␣%f␣and␣f(x)␣=␣%g␣\n␣', k, md, f0);
```

2　不动点迭代

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

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

$\varphi(x)$称为迭代函数，给定初值x_0，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。

例 2 改写上例

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

为

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

构造简单迭代格式，并观察收敛性

```
x0 = 3.5;
for k = 1:6
x0 = log(12*x0  +1) ./ log(3),
end
```

2.1 收敛性和误差估计

问题：迭代格式何时收敛？

定理 1 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

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

则 $\varphi(x)$在 $[a,b]$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

$$h(a) \ge 0, \quad h(b) \le 0$$

介值定理即得结论.

设x^*是唯一的不动点，则迭代格式产生序列 $x_k \in [a,b]$, 且

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

另外,

$$|x_k - x_{k-1}| = |\varphi(x_{k-1}) - \varphi(x_{k-2})| \le L|x_{k-1} - x_{k-2}| \le ... \le L^k|x_1 - x_0|, \ (\diamond)$$

对任意正整数p,

$$\begin{aligned}|x_{k+p} - x_k| &\le |x_{k+p} - x_{k+p-1}| + |x_{k+p-1} - x_{k+p-2}| + ... + |x_k - x_{k-1}|\\ &\le \left(L^{k+p} + L^{k+p-1} + ... + L^k\right)|x_1 - x_0|\\ &= \frac{L^k - L^{k+p}}{1-L}|x_1 - x_0|\end{aligned}$$

上式令 $p \to \infty$, 得到先验误差估计

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

实际计算过程中，由($\diamond$)式 得到

$$|x_{k+p} - x_k| \le |x_{k+p} - x_{k+p-1}| + ... + |x_k - x_{k-1}| \le \left(L^p + L^{p-1} + ... + 1\right)|x_k - x_{k-1}|$$

令 $p \to \infty$, 得到后验误差估计

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

2.2 局部收敛和收敛阶

定义 1 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:

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

对f(x) = 0 可以构造不同的迭代格式，收敛与否? 以及收敛速度可能不同。 李庆扬p218例4

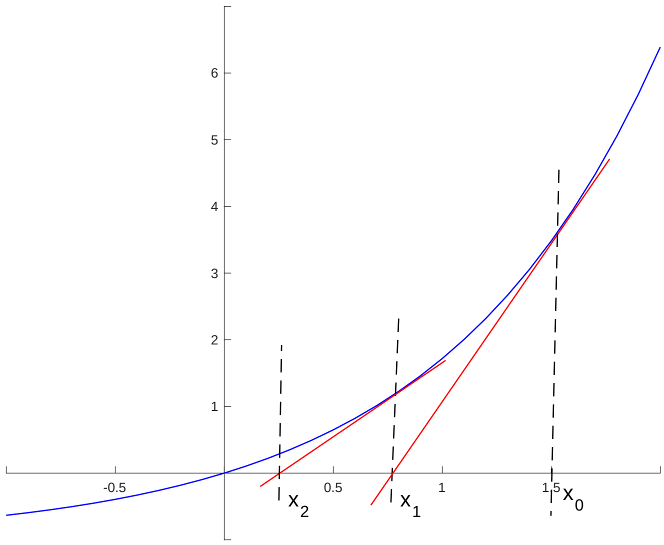
定义 2 若迭代格式$x_{k+1} = \varphi(x_k)$收敛于x^*， 记第k步迭代误差

$$e_k = x^* - x_k$$

若迭代误差满足

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

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.



3 Newton迭代(Raphson, Simpson) 及其变形

Newton法的几何解释

公式推导: 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

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

也可将迭代格式改写为如下形式(Newton法标准形式)

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

想一想: 每次迭代的主要运算量在哪里?

相关结论: 局部收敛, 且对于单根2阶收敛。

例 3 (作业) 设x^*是$f(x)$的零点, 若$f(x^*) \neq 0$, 证明: Newton迭代格式

$$x_{k+1} = x_k - \frac{f(x_k)}{f^{'}(x_k)}, \quad k = 0, 1, 2...$$

平方收敛.

提示：构造$\dfrac{\mid x_{k+1}-x^{*}\mid}{|x_k-x^{*}|^2}$的等价式 并利用Taylor公式，分析其收敛到常数.

例 4 构造

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

的牛顿迭代格式。

记

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

```
f = @(x) 3.^x - 1 - 12*x;
df = @(x) 3.^x.*log(3) - 12;
% ----
x0 = 3.5;
for k = 1:3
    s = - f(x0) / df(x0);
    x0 = x0 + s;
end
```

(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? Newton法对应的迭代函数φ是什么? 导数多大?

Tips: Newton法局部收敛，因此对初值要求较高；可以结合二分法来用。

另外Newton法对重根1阶收敛：例如：$f(x)=x^3=0$

3.1 割线法(弦截法)

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?

差商替代导数

利用

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

由Newton公式

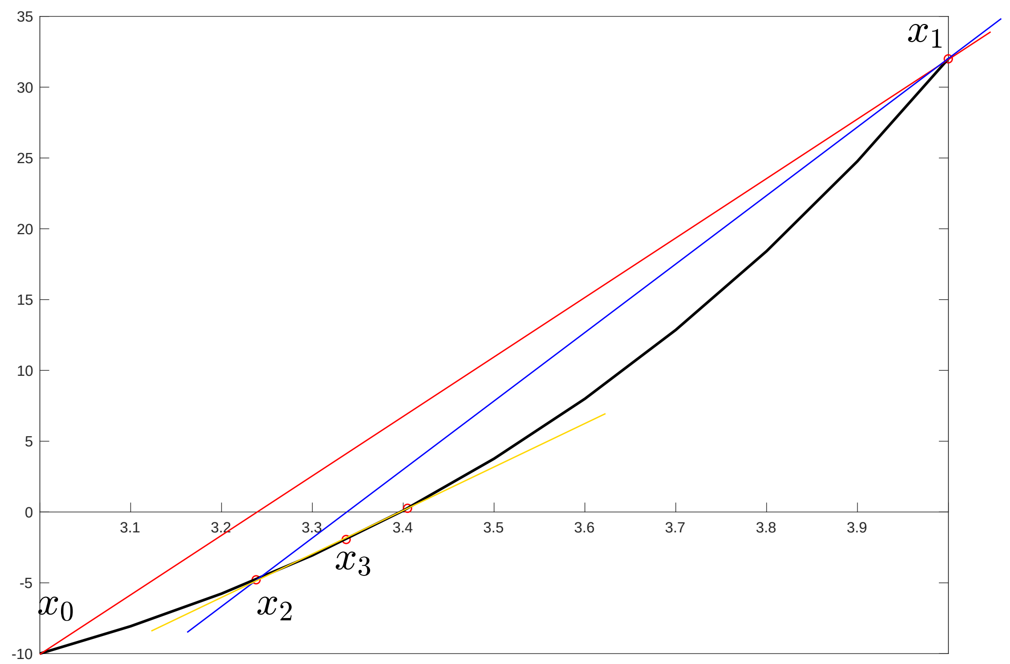
$$x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)}$$

可得

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

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

弦截法超线性收敛!! 演示 Matlab举例计算前例，效果也不错！



3.2 *Müller* 方法(抛物线法)

Newton法和割线法都是采用“以直代曲”思想，Muller方法的核心思想：以抛物线带曲线 假设已知三个近似零点x_{k-2}, x_{k-1}, x_x, 如何利用抛物线近似曲线$f(x)$, 并更新零点。

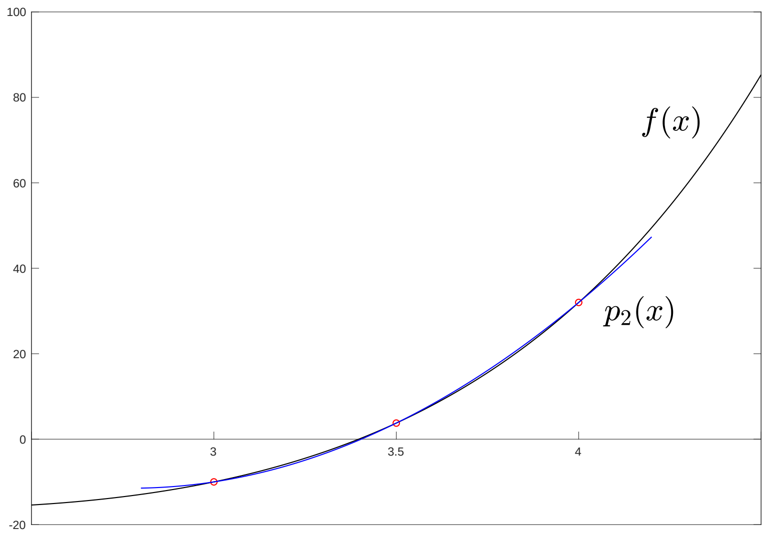
1. 构造二次插值多项式(Vandermonde矩阵)

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

2. 计算上式零点(有求根公式)，有两个 x_{k+1}, 还有可能出现复数

3. 挑一个合适的，继续更新

```
% muller method
% f = @(x)  (3.^x - 1) - 12*x;
% x0 = 3.3; x1 = 3.5; x2 = 3.6;
f = @(x) 16*x.^4 - 40*x.^3 + 5*x.^2 + 20*x + 6;
x0 = 0 ; x1 = 0.1; x2 = 0.2;
% ----
A = ones(3); maxI = 7; tol = 1e-9;
for kk = 1:maxI
```




```
y = [x0; x1; x2];
fy = f(y);
c0 = fy(1);
A(:,2:3) = [y, y.^2];
c = A \ fy;
% p = c0 + c1*x + c2*x^2;
c = c ./ c(3);
% -- p = c(1) + c(2)*x + x^2;
DELT = sqrt(c(2)^2 - 4*c(1));
z1 = (-c(2) + DELT) ./ 2 ;
z2 = (-c(2) - DELT) ./ 2 ;
%  update x0, x1, and choose x2
x0 = x1;  x1 = x2;
if abs(f(z1)) < abs( f(z2) ) % may be costly, but work
   x2 = z1;
else
   x2 = z2;
end
% --------
  if abs(f(x2)) < tol
      break
  end
end
x2,  abs(f(x2))
```

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

标注 1 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！！ 所有超线性收敛的方法均是高效的，且效率区别不大。例如：方法A是4阶收敛， 方法B是2阶收敛的；初始误差为0.1，则4阶方法2步迭代达到1e-16, 而2阶方法4步可到达1e-16.

3.3 Steffensen's method

即要Newton法的二阶收敛速度，又不要计算导数——Steffenson方法可以实现. Steffensen方法也可视为牛顿法的变形.

$$\begin{cases} x_{k+1} = x_k - \dfrac{f(x_k)}{\frac{f(x_k+f(x_k))-f(x_k)}{f(x_k)}} \\ x_0 = \text{ initial guess} \end{cases}$$

或者

$$\begin{cases} s = -\frac{f(x_k)^2}{f(x_k+f(x_k))-f(x_k)}, & \left(\Delta_1 = f(x_k),\ \Delta_2 = f(x_k+\Delta_1),\ s = -\frac{\Delta_1^2}{\Delta_2-\Delta_1}\right) \\ x_{k+1} = x_k + s; \end{cases}$$

上述算法需要计算两次函数值，不需要计算导数值. 在$f'(x)$计算麻烦时，可提高计算效率. 注意到 $\frac{f(x_k+f(x_k))-f(x_k)}{f(x_k)}$可视为导数的近似，但是当 初值$x_0$误差较大时，$f(x_0)$数值较大，此时收敛较慢. 可进行适当修正. 例如 选择（或结合）非精确Newton法：

$$f'(x_k) \approx \frac{f(x_k+h)-f(x_k)}{h}$$

读者可测试以下代码的收敛速度，若直接使用steffensen算法效率较慢.

```
 f = @(x) 3.^x - 1 - 12*x;
x0 = 3.5;  h_tol = 1e-5;
% % steffensen meth/ inexact newton
for k = 1:5
    d1 = f(x0);
    if abs(d1) > h_tol
        h = h_tol;
    else
        h = d1;
```

```
    end
    d2 = f(x0+h);
    s =  -d1*h /(d2 - d1) ;
    x0 = x0 + s;  f(x0),
end
```

4 多项式零点的补充说明

任意的多项式求零点可以转化为首一(monic) 的多项式

$$Q_n(x) = x^n + c_1x^{n-1} + c_2x^{n-2} + ...c_{n-1}x + c_n = 0. \tag{2}$$

的零点计算. 若给出合适初值，Newton法、割线法、抛物线法（可求复根）均可计算.

更为常用的方法是借助于计算友矩阵的特征值，得到多项式的根. 该方法的优点是稳定性好，一次可以计算出 所有根，且对初值没有要求.

例 5 验证(2)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。矩阵C 称为$Q_n(x)$ 的友矩阵(Companion Matrix). 也称为多项式$Q_n(x)$ 的线性化.

提示: 对行列式$\det(\lambda I - C)$按行或者按列展开，检验即可。 若求出矩阵 C 的特征值[1]， 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：求其友矩阵(companion matrix)的特征值!

标注 2 数值分析有其他手段求矩阵特征值！！！（例如：幂法、反幂法、QR 算法）. Matlab 函数 roots 用于计算 多项式的根，采用的算法即是将多项式转化为友阵，再用QR迭代计算友阵的所有特征值. 这里也再次提醒读者体会“理论数学”和“计算数学”的异同！！！ 更一般的情况，若$Q_n(x)$以Chebyshev多项式或者Legendre多项式为基函数展开，其相应的“友矩阵”称为 colleague 或者comrade矩阵 也可表示为Hessenberg矩阵，进而利用稳定的QR算法计算其特征值， 进一步阅读 可参考[2].

[1]矩阵的特征值计算有其它更有效的方法，例如QR算法

[2]Nakatsukasa Yuji, Noferini V, On the stability of computing polynomial roots via confederate linearizations, Mathematics of Computation, (2015), 2391-2425, 85(301) 或者短文 roots of cheby series

5 非线性方程组的Newton迭代法

例 6 给定二元函数 $g(x,y)$, 求$g(x,y)$ 在 区域$\Omega\subset\mathbb{R}^2$ 内的驻点（极小值，极大值，鞍点）.

考虑二阶方程组 <u>(n阶方程组可类似分析)</u>

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

上式可以写成向量形式:

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

类似可以考虑n个未知量，n个非线性方程的情形。

标注 3 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。这里只介绍Newton法。

回顾单变量的Newton法: 给定问题$f(x)=0$; 构造迭代格式

$$x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)}\Leftrightarrow\begin{cases} s=-\dfrac{f(x_k)}{f'(x_k)}, \\ x_{k+1}=x_k+s. \end{cases}$$

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

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

则非线性方程组的Newton迭代格式如下

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

实际计算时不求逆矩阵，而是求解如下线性方程组

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

的等价形式

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

上述过程，那部分计算量最大?

例 7 计算$\boldsymbol{F}(\boldsymbol{x}) = \mathbf{0}$, 取$x_0 = [0,0]^{\mathrm{T}}$

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

```
% ===== Newton for nonlinear system
x0 = [0; 0];
for k = 1:4
    A = df1(x0); b = - f1(x0);
    s = A \ b;
    x0 = x0 + s;
end
x0, f(x0)
% =====  F(x)
function y = f1(x)
x1 = x(1) ; x2 = x(2);
y(1) = x1^2 - 10*x1 + x2^2 + 8;
y(2) = x1*x2.^2 + x1 - 10*x2 + 8;
y = y(:);
end
% --- F'(x)
function J = df1(x)
x1 = x(1) ; x2 = x(2);
%
J = zeros(size(x));
J(1,1) = 2*x1 - 10; J(1,2) = 2*x2;
J(2,1) =  x2.^2 + 1; J(2,2) = 2*x1*x2 - 10;
end
```

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

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 > 百万？），可以放弃精确Newton法。

5.1 Inexact Newton method(略)

Algorithm 2 Inexact Newton method

1: Set initial guess x_0

2: for k = 1,2,3,... do

Find a vector$s$ satisfied

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

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

实用的处理方式，就是避免计算Jacobian, 而寻找近似替代品，例如有限差分

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

注意到第二步（红色部分），本质上是求解线性方程组 $As = b$, 而第5章我们介绍了Arnoldi方法构造Krylov子空间 $\boldsymbol{K}_n(A, b)$, 得到

$$AQ = QH \quad 以及 \quad A = QHQ^{\mathrm{T}}$$

求解线性方程组$Ax = b$转化为求解$QHQ^{\mathrm{T}}x = b$. 而产生Q, H 的过程只需要计算 矩阵(算子)A乘以向量得到新的向量, 这一过程可有差分近似得到！ Good！

5.2 差分近似导数矩阵 (略)

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

例 8 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。

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

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

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

Matlab演示：中心差分效率更高！Diff_jacobi.m

取定差分步长(例如$\sigma = 1e-5$) 结合FOM(Arnoldi full orthogonal method) 方法求解近似方程组

$$As = b \to \mathcal{L}s = b$$

可得 s.

```
% ====== inexact Newton method
x0 = [0;0];  SIGMA = 1e-4;  tol = 1e-6;
for k = 1:4
    b = - f1(x0);
%  ---- %  approx   s = A \ b;
    s = DiffArnoldi1(@f1,x0,SIGMA,b);
% --------
    x0 = x0 + s;
end
x0, f1(x0),k
```

```
% =====  vector funciton, F(x)
function y = f1(x)
x1 = x(1) ; x2 = x(2);
y(1) = x1^2 - 10*x1 + x2^2 + 8;
y(2) = x1*x2.^2 + x1 - 10*x2 + 8;
y = y(:);
end
```

```
% ====== solve Ax = b, by approx (FOM) krylov subspace
function x0 = DiffArnoldi1(F,p,SIGMA,b)
% - finite diff  approx Jacobian, F'(p)*x = b; FOM for Ax = b;
m = size(b,1); V = zeros(m,m+1); H = zeros(m+1,m);
%  ----
     r0 = b;  bet = norm(r0);  % initial guess is ZEROS
     b0 = zeros(m,1); b0(1) = bet;
     v1 = r0./bet;  V(:,1) = v1;
    for j = 1:m
%          w = A*r0
        w = F(p + SIGMA*V(:,j)) - F(p);  w = w ./ SIGMA;
        for i = 1:j
            h = dot(w,V(:,i));
            w = w - h*V(:,i);
            H(i,j) = h;
        end
        H(j+1,j) = norm(w); V(:,j+1) = w ./ H(j+1,j);
    end
    y =  H(1:m,:) \ b0;  x0 = V(:,1:m)*y;
```

```
end
```

标注 4 上述算法可以避开Jacobi矩阵的计算。理解上述求解过程，需要读者理解 Arnoldi正交化过程（见第六章讲义最后）。 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，FOM（Full Orthogonal Method）完全可以接受。

6 Newton法的应用: 多元函数的驻点

给定二阶可导的多元函数$f(\boldsymbol{x})$, $\boldsymbol{x} \in \boldsymbol{R}^n$, 则其驻点满足$\nabla f(\boldsymbol{x}) = \mathbf{0}$. 利用 Newton法可得

1. 给定初值$\boldsymbol{x}_0$, Hessian矩阵 H_f,
2. 按照下式

$$\begin{cases} H_f(\boldsymbol{x}_0)\boldsymbol{s} = -\nabla f(\boldsymbol{x}_0) \\ \boldsymbol{x}_1 = \boldsymbol{x}_0 + \boldsymbol{s} \end{cases}$$

   更新$\boldsymbol{x}$即为求多元函数驻点的Newton法。

例 9 (研究生留做作业) Let $f : \boldsymbol{R}^n \to \boldsymbol{R}$ be given by

$$f(x) = \frac{1}{2}x^T Ax + x^T b + c,$$

where $A \in \boldsymbol{R}^{n\times n}$ is symmetric positive definite, $b \in \boldsymbol{R}^n$, and $c \in \boldsymbol{R}$.

1) Show that Newton's method for minimizing this function converges in one iteration from any starting point $x_0 \in \boldsymbol{R}^n$.

2) If the steepest descent method is used on this problem, what happens if the starting value x_0 is such that $x_0 - x_*$ is an eigenvector of A, where $x_* \in \boldsymbol{R}^n$ is the solution?

7 小结

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; 但是将这一步骤修改为Krylov子空间方法(如 gmres方法)。Inexact Newton method依然有效。

上机练习

1. 结合二分法与Newton法求解下列方程的实根

(1) $x^2 - 3x + 2 - e^x = 0$

(2) $x^3 + 2x^2 + 10x - 20 = 0$

2. 分别用二分法，Newton法，割线法求解

$$xe^x - 1 = 0$$

3. 利用Newton法或者Inexact Newton method (个人更推荐该方法) 计算

$$\begin{cases} 3x_1 - \cos(x_2x_3) - \dfrac{1}{2} = 0 \\ x_1^2 - 81(x_2 + 0.1)^2 + \sin x_3 + 1.06 = 0 \\ e^{-x_1x_2} + 20x_3 + \dfrac{10}{3}\pi - 1 = 0 \end{cases}$$