高等数值算法与应用(3)

－ 非线性方程求根与局部极值 (chap4)－

喻文健

Outline

- 二分法
- 牛顿法、割线法
- 逆二次插值法
- Zeroin算法
- 求局部极值与最优化

非线性方程基本理论

- 何谓非线性方程?

$$f(x) = 0$$

- 一般地，解的存在性和个数很难确定

(1) $e^x + 1 = 0.$ **无解.**

(2) $e^{-x} - x = 0.$ 有**一个解.**

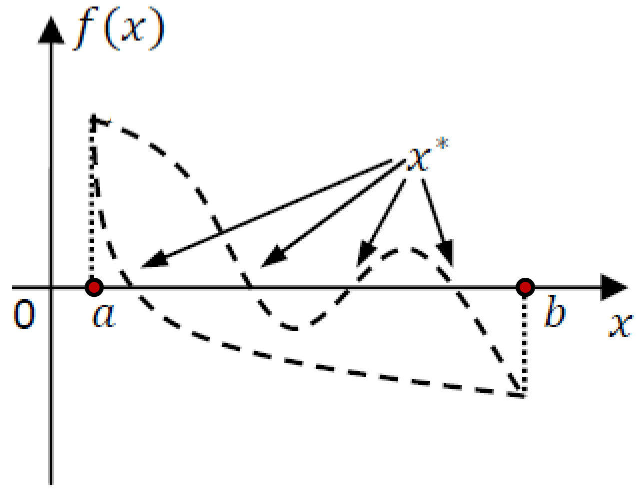
(3) $\cos x = 0.$ 有**无穷多个解.**

(4) $x^3 - 6x^2 + 5x = 0.$ 有**三个解.**

方程的根, 即函数$f(x)$的零点

- 实际问题中，一般求某个范围(区间)内的解，采用迭代解法 (与优化问题关系密切)
- **m重根**x^*: $f(x^*) = f'(x^*) = \cdots = f^{(m-1)}(x^*) = 0$

二分法



- **定理**：若$f(x) \in C[a,b]$，且$f(a)f(b) < 0$，则区间(a,b)内至少有一实根
- 称(a,b)为有根区间
- 根据$f((a+b)/2)$的正负号, 可得长度减半的第2个有根区间. ..., 如此反复, 第k个有根区间中点x_k逼近解

$$|x_k - x^*| < (b_k - a_k)/2 = (b_0 - a_0)/2^{k+1} \xrightarrow{k\to\infty} 0$$

- 估计达到特定准确度所需的二分次数, 即**计算量**
- 浮点算术体系: 寻找非常小的区间(可能是相邻的浮点数)，使得在这个区间上函数值的正负号发生改变

二分法

- 例:求解方程$f(x)=x^2-2=0$，初始区间为[1, 2].
- 程序迭代52次后结束
- 打印出的区间端点值

… …

a = 3ff6a09e667f3bc8 (format hex)

a = 3ff6a09e667f3bcc

b = 3ff6a09e667f3bce 相邻浮点数

b = 3ff6a09e667f3bcd 死循环!

- 放宽while条件，多迭代几步如何？
- 已达最准情况：有根区间长为$2^E \cdot$eps
  E为准确解x^*的**指数**，即$\lfloor \log_2 |x^*| \rfloor$
  双精度下，解的误差下限:2^{E-52}

```
a = 1
b = 2
k = 0;
while b-a > eps
   x = (a + b)/2;
   if x^2 > 2
      b = x
   else
      a = x
   end
   k = k + 1;
end
```

接近浮点数表示的最高准确度

二分法

- Matlab程序

bisect.m

```
k = 0;
while abs(b-a)>eps*abs(b)
  x = (a + b)/2;
  if sign(f(x))==sign(f(b))
    b = x;
  else
    a = x;
  end
  k = k + 1;
end
```

(总迭代52步?)

- 只要准确解≠0，可求得几乎最准确的结果
  - 若f(x)=0呢? 准确解为0呢?

赋值a，循环并不停

死循环！因为a, b符号不同, abs(b−a)≥abs(b)>0

改进程序

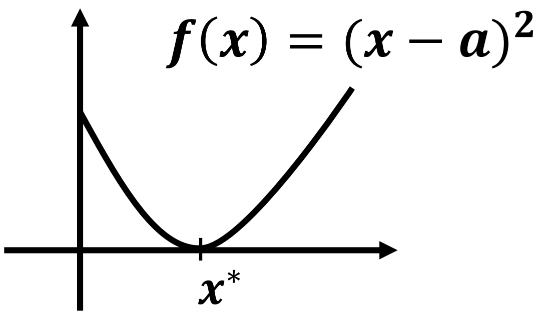
```
function [x, k]= my_bisect(f, ab)
k = 0; a=ab(1); b=ab(2);
fb= f(b);
while abs(b-a) > eps*abs(b)
  x= (a + b)/2;
  fx= f(x);          //计算效率
  if fx==0           //特殊情况
    break;
  elseif sign(fx)==sign(fb)
    b = x;   fb= fx;
  else
    a = x;
  end
  k = k + 1;
end
```

tol*x_est

不会死循环了!

小结

- 二分法是求单个方程$f(x) = 0$的实根的可靠算法，总能收敛.
- 在实际计算时，解的准确度有极限(~浮点数系统).
- 缺点是：
  - 收敛速度较慢
  - 需要两个初始值(有根区间)
  - 无法求偶数重的重根
  - 对含多个根的区间，求出哪个根比较随机



函数作为问题数据

- Matlab程序的使用 -- 函数作为输入参数
  - 匿名函数：
    ```
    >> f = @(x) x-1-1/x;
    ```
  - .m文件定义函数，引用时写@*fun_name*即可
- 带参数的函数(主要/次要自变量) $f(t,z,w)=t^{z-1}(1-t)^{w-1}$

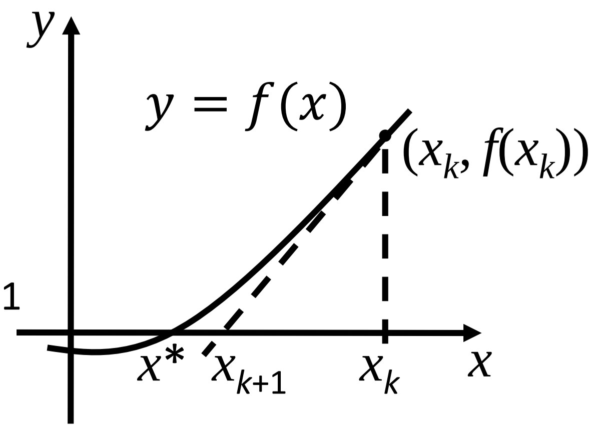
  z, w为参数
  - f = @(t,z,w) t^(z-1)*(1-t)^(w-1)
  - .m文件定义的函数（第1个参数是主要自变量)
- 一般非线性方程求根程序的接口
  - fzerotx.m: b = fzerotx(F,ab,varargin)

可变数量的变量，存次要变量的值

牛顿法、割线法

牛顿法

- 原理
  - 利用曲线上点$(x_k, f(x_k))$处的切线，求x_{k+1}



- 计算公式

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

- 程序my_Newton

```
k =0;
while abs(x- xprev) > tol*abs(x)
    xprev= x;
    x = x - f(x)/fprime(x);
    k = k+1;
end
```

(f(x)=0时退出循环)

- 例：计算$\sqrt{2}$，$f(x)=x^2-2$
  - 若$x_0=1$，仅需6步迭代达到最准结果

若对于收敛快的算法，是误差的估计

若tol=eps，基本上达最准

若准确解为0? 一般没问题，因为x可能=xprev

其他优点：可以算复数根、解方程组

牛顿法的问题

课后思考

- 对很光滑的$f(x)$,且初值接近解, $e_{k+1} = \frac{f''(\xi)}{2f'(x_k)} e_k^2 = O(e_k^2)$
- 若$f(x)$不光滑，可能不收敛

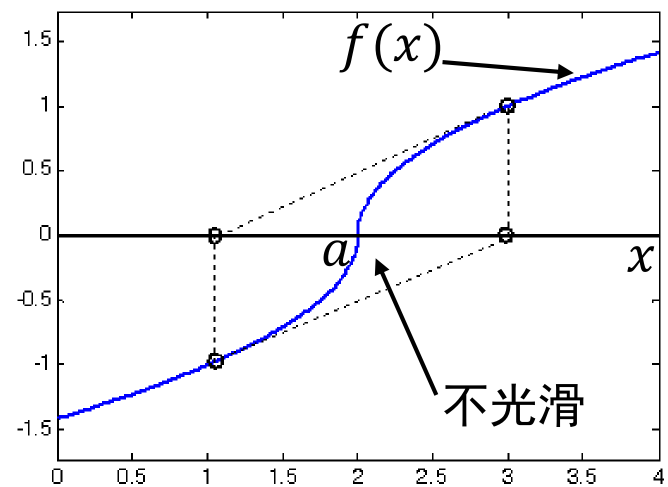
$$f(x) = \text{sign}(x-a)\sqrt{|x-a|} = 0$$

$$x_{k+1} = x_k - 2(x_k - a)$$

$$\Longrightarrow x_{k+1} - a = -(x_k - a)$$

迭代解在$x = a$点左右跳动

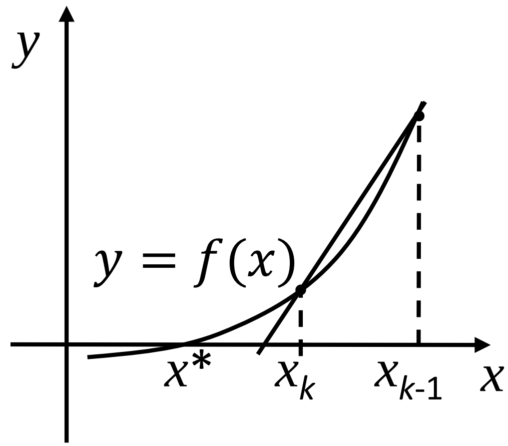
对任意初值都不收敛的例子



另两个缺点:

- 需要计算导数$f'(x)$
- 要求初值接近准确解

割线法



- Secant method (quasi-Newton)
  - 牛顿法中的导数计算替换成近似导数
  - 需要两个初始值x_0, x_1, 后续解为

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

- 程序my_secant

```
k =0;
while abs(b- a) > eps*abs(b)
    c= b;              % b是x的近似解
    b = b + (b-a)/(f(a)/f(b)-1);
    a = c;             % a是上一个近似解
    k = k+1;
end
```

- 例: 计算$\sqrt{2}$, $f(x)=x^2-2$
  - ➢ 若初始值为1, 2, 仅迭代7步
- 超线性收敛速度:

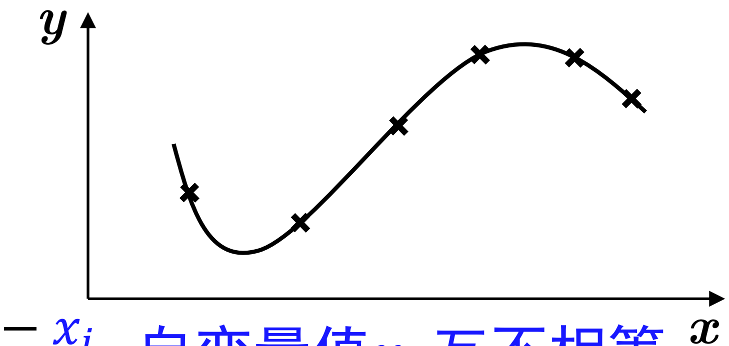
$$e_{k+1} = O(e_k^{\phi})$$
$$\phi \approx 1.618$$

缺点: 需要两个较好的初值

逆二次插值法

逆二次插值法

- 多项式插值
  - 通过离散数据点的多项式函数
  - Lagrange插值



$$L_n(x)=\sum_{k=0}^{n} y_k l_k(x)\ ,\ l_k(x)=\prod_{j\neq k}\frac{x-x_j}{x_k-x_j}$$

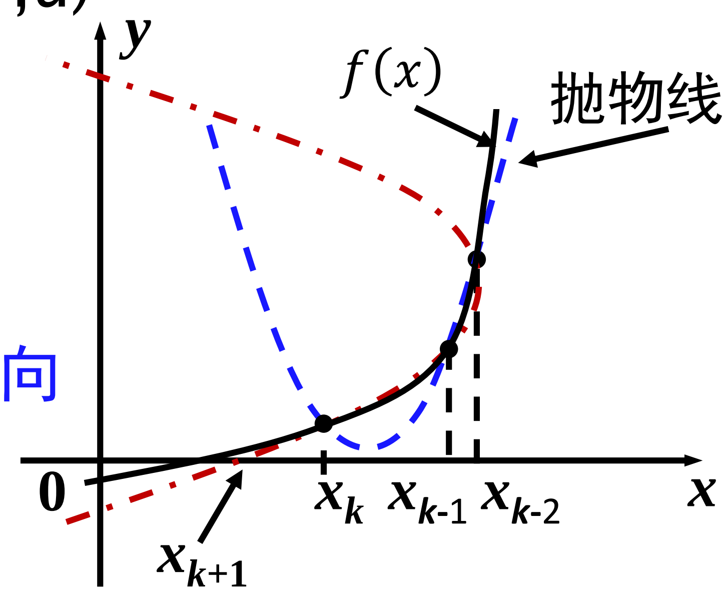
自变量值x_k互不相等

  - NCM实现：v=polyinterp(x,y,u)

也用polyfit, polyval算

- 二次插值/逆二次插值法
  - $x_k, x_{k-1}, x_{k-2} \xrightarrow{\text{二次多项式近似}f(x)} x_{k+1}$
  - 但抛物线与横轴可能无交点!
  - 逆二次插值: x看成y的函数(侧向抛物线)

要求y_{k-2}, y_{k-1}, y_k不同



逆二次插值法

设a, b, c为三个近似解x_{k-2}, x_{k-1}, x_k

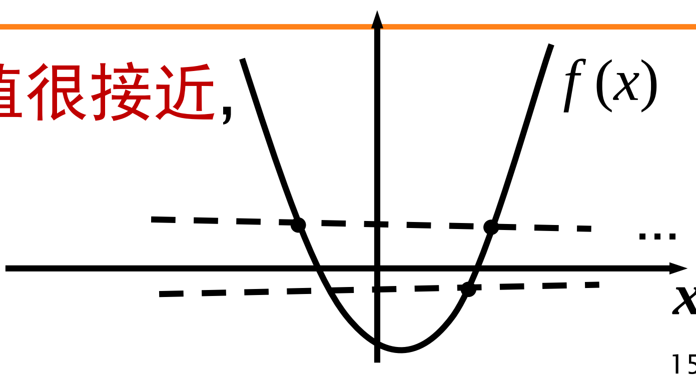
$$a = P(f(a)),\ b = P(f(b)),\ c = P(f(c))$$

- 插值函数P满足:
- 则$x_{k+1} = P(0)$ 是它与x轴交点
- 程序

IQI算法(Inverse quadratic interpolation)

```
k =0;
while abs(c- b) > eps*abs(c)
   x=polyinterp([f(a), f(b), f(c)], [a, b, c], 0)
   a= b;
   b= c;
   c= x;
   k = k+1;
end
```

- 优缺点
  - 收敛速度快
    $e_{k+1} \approx O(e_k^{1.839})$
  - $f(x_{k-2})$, $f(x_{k-1})$, $f(x_k)$虽不等, 但值很接近, 得到的x_{k+1}将远远偏离求解区间
  - 割线法也有此不稳定性 (斜率~0)



Zeroin算法

通用求根算法zeroin

- 求解$f(x) = 0$的方法比较

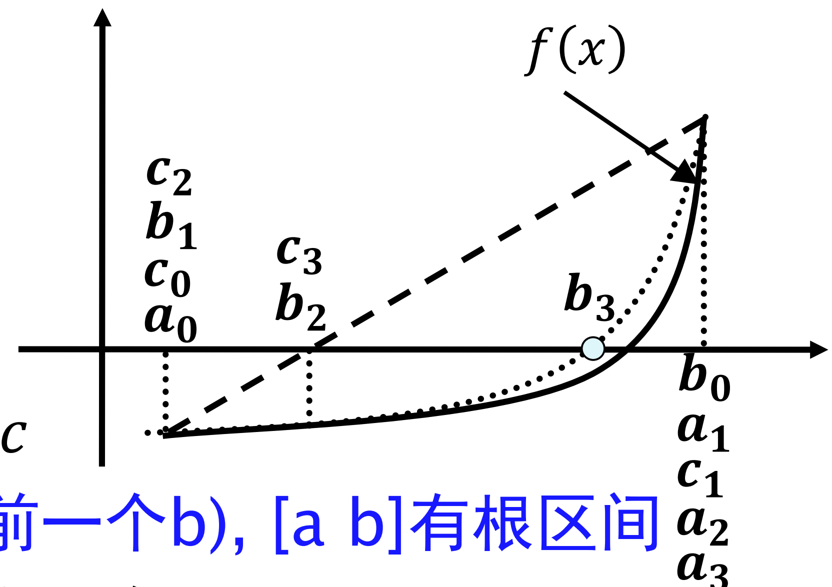
习题 4.8

  - 割线法/IQI法: 局部收敛(依赖初值选取)、函数的光滑性要求(虽然比牛顿法的低)；收敛速度快
  - 二分法: 全局收敛(易选初值)、仅需函数连续；收敛慢
  - 将两者结合得到稳定、快速算法，也叫Brent方法(1973)
- Zeroin算法主要步骤
  - 输入有根区间$[a, b]$, 迭代中维护a, b, c三个近似解
  - 保证b为当前最优解，a与它构成有根区间，而c为次优解
  - 重复下面的步骤，直到$|f(b)|$足够小或$|a - b|$足够小
    - 执行: 逆二次插值法/割线法/二分法; 调整a, b, c的值

通用求根算法zeroin

- 每步迭代计算都先做调整
  - ①若新算出的$f(b)$正负号与$f(a)$相同，将c的值赋给a
  - ②若$|f(a)|<|f(b)|$，则对调a, b的值，然后将a的值赋给c
  - **保证**: b是最优解, c次优解(前一个b), [a b]有根区间

纠正解可能的偏离



- 执行IQI/割线法/二分法中的哪个?
  - 若$c \neq a$，用a, b, c及其函数值做IQI法一步，否则割线法
  - 若IQI法/割线法得到的解<u>足够满意</u>(相邻解之差的缩小程度、位置在有根区间内)，用它更新b，否则执行一步二分法, 更新c

通用求根算法zeroin

- 算法的特点
  - 每步迭代都使有根区间缩小
  - 抛弃逆二次插值/割线法得到的”不满意”解
  - 按”逆二次插值, 割线法, 二分法”的优先顺序生成下一步解，保证较快的收敛速度
  - 算法稳定、通用性强，是Matlab命令fzero的基础
- NCM中的演示程序
  - fzerogui

缺省f(x)=x^3-2x-5

```
>> fzerogui
>> fzerogui(bessj0, [0 3.83])
```

函数bessj0=@(x) besselj(0,x);

fzero命令

(求单变量连续函数的零点)

>> opt= optimset('tolX', 1e-2);

- 语法格式

结构 optimset设置结构

  - [x,fval,exitflag,output] = fzero(fun, x0, opt)
  - fval为解x对应的f(x); exitflag=1表示正常，其他出错
  - 若x0为标量，返回它附近的解 (先自动找有根区间)
  - 若x0为数组，则x0(1:2)为有根区间，否则报错
  - output返回迭代次数等信息, opt设置收敛条件, 显示等
  - 其他参数放最后(若有)
  - 例: 第1类零阶贝塞尔函数$J_0(x)$
  - 求它的前10个零点

```
>> bessj0= @(x) besselj(0,x);
>> ezplot(bessj0, [0, 10*pi]);
for n=1:10
   z(n)= fzero(bessj0, [(n-1) n]*pi);
end
hold on; plot(z, zeros(1, 10), 'o')
```

fzerotx程序 (fzero的简化版本, 无复杂输入输出)

- 程序停止的判据
- 割线法与IQI

```
m = 0.5*(a - b);
tol = 2.0*eps*max(abs(b),1.0);
if (abs(m) <= tol) | (fb == 0.0)
   break
end
```

f(b)恰好=0，或者有根区间长度为舍入误差量级(绝对、相对)

```
s = fb/fc;
if (a == c)
   p = 2.0*m*s;     q = 1.0 - s;
else
   q = fc/fa;    r = fb/fa;
   p = s*(2.0*m*q*(q -r) -(b -c)*(r - 1.0));
   q = (q - 1.0)*(r - 1.0)*(s - 1.0);
end;
if p > 0, q = -q; else p = -p; end;  p总>0
if (2.0*p < 3.0*m*q - abs(tol*q)) & (p < abs(0.5*e*q))
   e = d;  d = p/q;
else
   d = m;  e = m;
end;
```

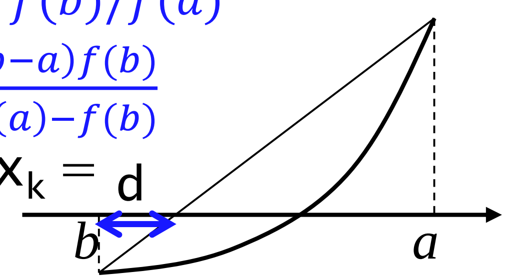
割线法:

$p = (a-b)f(b)/f(a)$

$q = 1 - f(b)/f(a)$

$$-\frac{p}{q} = \frac{(b-a)f(b)}{f(a)-f(b)}$$

$= x_{k+1} - x_k = d$



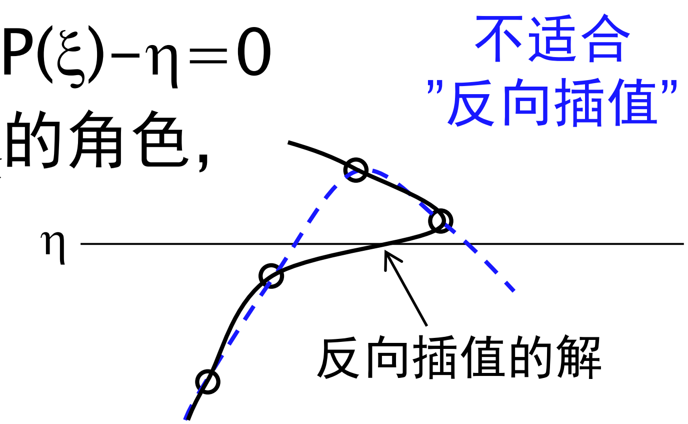
解在有根区间内;

$\left|\frac{p}{q}\right| < \frac{3}{4}$有根区间长度，且相邻解之差缩一半

(满足一定条件才接受割线/IQI法的新解)

相关的问题

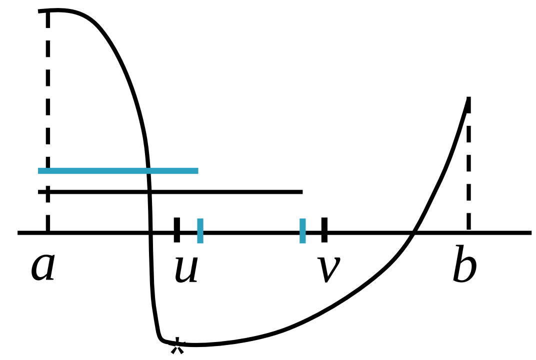
- 寻找函数为某个值的解
  - 给定一个函数F(x)和值η，求ξ使得F(ξ)=η
  - 解：令f(x)= F(x)−η, 再用非线性方程求解器
  - 若函数F(x)是由数据点(x_k, y_k)表示的呢？
  - 对数据点插值得到P(x)，然后求解P(ξ)−η=0
  - 有时"反向插值"更方便, 颠倒x_k和y_k的角色, 仅需对插值函数Q(y)求值, ξ=Q(η)
  - 前提: y_k序列在η附近是单调的



- 其他问题
  - 多项式方程求所有根: >> r = roots(p)
  - 非线性方程组求根: x= fsolve(fun,x0); 自编(拟)牛顿法

求局部极值与最优化

最优化与fminbnd



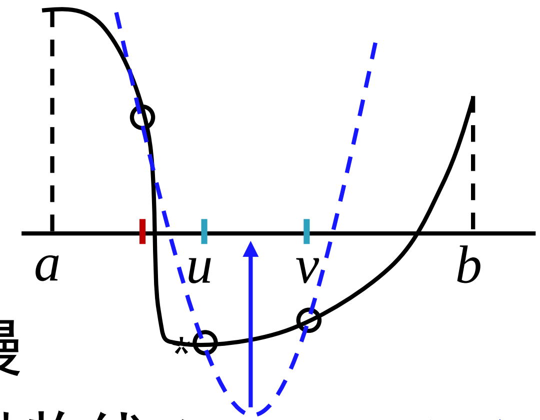
- 求单变量连续函数局部极小值
  - 函数形式复杂或表达式未知
  - 区间[a, b]内只有一个极小值点 (否则随意地找到某一个)
- 求解方法　　　　($\overrightarrow{uv}$段函数值上升, v以外的不考虑)
  - 三等分法：若f(u)<f(v), 则极小值在区间[a, v]内，否则极小值在[u, b]内. 逐次使区间缩小
  - 每次算2个函数值，区间约按0.667比例缩小　　少算一次函数!
  - 未必要三等分! 使u在[a,v]中位置等同于[a, b]中的v?
  - 设$v-a=\rho$, 则由对称性, $u-a=1-\rho$ (设[a, b]长为1)

$$\rho = \frac{1-\rho}{\rho} \Longrightarrow \rho = \frac{\sqrt{5}-1}{2} \approx 0.618 = \frac{1}{\phi}$$

黄金分割搜索!

搜索区间缩小得快!
计算量少!

最优化与fminbnd



- 更好的算法
  - 黄金分割搜索很稳定，收敛速度仍慢
  - 在区间内有三个不同点，可插值出抛物线(如函数值"高–低–高"时)，用其极小值点作为近似解 与黄金分割搜索结合起来！
  - 抛物线解的接受条件：在区间内，相邻解间距缩小，等

类似于zeroin算法

- fminbnd
  - Matlab命令fminbnd，求单变量函数局部极小值
  - [x,fval,exitflag,output] =fminbnd(fun,x1,x2,opt)
  - x1<x2，为区间两端点；fval为目标函数(极小)值
  - exitflag退出标记，output包含算法、函数求值次数等
  - 用optimset设置opt中选项的值 (同fzero)

NCM中的fmintx程序

fminbnd的简化版本

u = fmintx(F,a,b,tol,varargin)

- 程序停止的判据
- 黄金分割搜索与抛物线法

```
while abs(x-xm) > tol
```

近似极小点

含极小值区间的中点

```
r = (x-w)*(fx-fv);
q = (x-v)*(fx-fw);
p = (x-v)*q-(x-w)*r;
q = 2.0*(q-r);
if q > 0.0,  p = -p; end
q = abs(q);
r = e;       e = d;
% Is the parabola acceptable?
para = ( (abs(p)<abs(0.5*q*r)) &
(p>q*(a-x)) & (p<q*(b-x)) );
if para
    d = p/q;
end
```

根据x, w, v这三点数据造抛物线

p, 或q取相反数

x+p/q在区间[a, b]内，且相邻解间距缩小

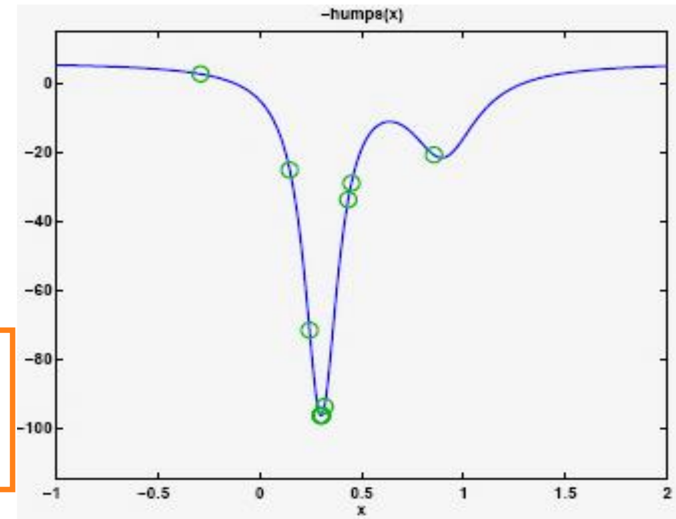
感兴趣的同学可做习题4.18, 4.20, 4.21

最优化与fminbnd

- fminbnd/fmintx的应用例子
  - 求−humps(x)的极小值

$$h(x) = \frac{1}{(x-0.3)^2+0.01} + \frac{1}{(x-0.9)^2+0.04} - 6$$

```
>> F= @(x) -humps(x);
>> fmintx(F, -1, 2, 1.e-4);
```



  - 搜索过程显示 (习题4.17)

step	x	f(x)
init:	0.1458980337	-25.2748253202
gold:	0.8541019662	-20.9035150009
gold:	-0.2917960675	2.5391843579
para:	0.4492755129	-29.0885282699
para:	0.4333426114	-33.8762343193
para:	0.3033578448	-96.4127439649
gold:	0.2432135488	-71.7375588319
para:	0.3170404333	-93.8108500149
para:	0.2985083078	-96.4666018623
para:	0.3003583547	-96.5014055840
para:	0.3003763623	-96.5014085540
para:	0.3003756221	-96.5014085603

fminbnd缺省的tolX是1e−4
但结果与fmintx的不一样

opt=optimset('tolX', 1e−6)

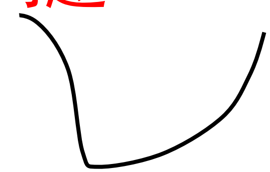
试试?

其他优化问题与算法

- 优化(规划)问题基本概念

$$\begin{cases} \min\ f(\boldsymbol{x})\,; \\ \text{s. t:}\ g_i(\boldsymbol{x}) \le 0,\quad i = 1, 2, \dots, m\ , \\ \qquad h_j(\boldsymbol{x}) = 0,\quad j = 1, 2, \dots, l\,. \end{cases}$$

  - 目标函数, 可行域, 约束条件
  - 局部最优, 全局最优; 凸集, (下)凸函数, 凸规划问题



  - 单变量约束优化问题 (fminbnd)

一维搜索方法
- 黄金分割法(区间收缩)
- 牛顿法/抛物线法(函数逼近)

  - 多变量无约束优化问题　求某点附近局部最优

搜索方法
- 定搜索方向
  - 只用函数值的直接搜索(方向轮替)
  - 用导数值的解析法(SD/Newton/CG)
- 定搜索步长: 各种一维搜索方法　(直接搜索法)

Matlab中的fminsearch使用Nelder-Mead单纯型法

  - 其他问题: 约束优化, 线性规划, 整数规划, …

Optimization Toolbox

符号运算工具箱

- 解析求解方程的solve命令
  - 例：求解方程 $\frac{\phi-1}{1}=\frac{1}{\phi}$
  - 得到的r为符号变量数组
  - 转变为数值: double, vpa
- 其他命令
  - 对多项式函数poly2sym
  - 用factor做因式分解

```
>> r = solve('x-1=1/x')
>> phi = r(1)
```

```
>> phi = double (phi)
>> vpa(phi, 50)
```

```
>> p= [1 -2 -3];
>> p= poly2sym(p)
>> factor(p)
```