M05M11084 最优化理论、算法与应用

5 无约束优化方法 I

无约束优化方法 I

参考：

1. 应用最优化方法及MATLAB实现，第5章，刘兴高 胡云卿
2. 最优化导论，第8~11章， Edwin K.P. Chong, Stanislaw H. Żak 著, 孙志强等译
3. 最优化基础理论与方法，第3章, 无约束优化方法，王燕军，梁治安，崔雪婷
4. Numerical optimization, Chapter 5,6,8, Jorge Nocedal Stephen J. Wright

1.引言

2.梯度下降法

3.牛顿法

引言

- 多变量优化问题是普遍问题
- 多维无约束优化问题的描述

$$\min f(\boldsymbol{x})$$
$$\boldsymbol{x}^* = \operatorname{argmin} f(\boldsymbol{x})$$

$f: \mathcal{R}^n \to \mathcal{R}$， $\boldsymbol{x} \in \mathcal{R}^n$， $f \in C^1$

- 多维无约束优化的方法分类

- 直接法
  - 线性最小二乘法
  - 坐标轮换法
  - 共轭方向法
    - GS
    - Powell方法
    - Powell改进
  - Downhill Simplex
  - 模式搜索
- 一阶导数
  - 梯度下降法
    - 固定步长梯度下降法
    - 最速梯度下降法
  - 共轭梯度法
    - FR
    - PRP
- 二阶导数
  - 牛顿法
  - 阻尼牛顿法
  - 拟牛顿法
    - DFP
    - BFGS
    - LBFGS
  - 高斯牛顿法
  - LM
- 信赖域
- 全局优化法
  - 遗传算法、模拟退火、粒子群

迭代算法的终止准则

$$\left\|\nabla f\left(\boldsymbol{x}^{k+1}\right)\right\| < tol$$

$$\left\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^{k}\right\| < tol \cdot \left[\min\left\{1, \left\|\boldsymbol{x}^{k}\right\|\right\}\right]$$

$$\left|f\left(\boldsymbol{x}^{k+1}\right) - f\left(\boldsymbol{x}^{k}\right)\right| < tol \cdot \left[\min\left\{1, \left|f\left(\boldsymbol{x}^{k}\right)\right|\right\}\right]$$

1.引言

2.梯度下降法

- 最速下降法
- 一般梯度下降法

3.牛顿法

等值线、梯度

$f: \mathcal{R}^n \rightarrow \mathcal{R}, \boldsymbol{x} \in \mathcal{R}^n$, f differentiable

level curve (set) $\{\boldsymbol{x} | f(\boldsymbol{x}) = c\}$, c constant

$$f(x^0) = c$$

$\nabla f(x^0) \neq \mathbf{0}$

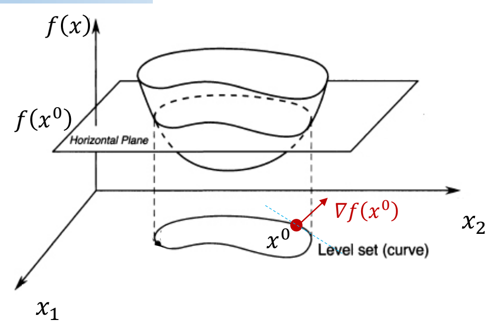
$\nabla f(x^0) \perp$ the tangent vector of $f(\boldsymbol{x}) = c$ at x^0

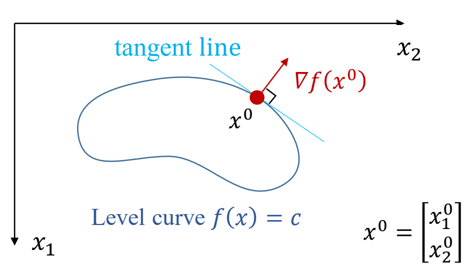
$$\varphi(t) = f\big(x(t)\big) = c \qquad x^0 = x(t_0)$$

$$\varphi'(t) = Df\big(x(t)\big)Dx(t) = 0$$

$$\nabla f(x)^T \dot{x}(t) = 0$$

For a given small displacement,
f increases more in the direction of the gradient than in any other direction.





负梯度方向的下降率最大

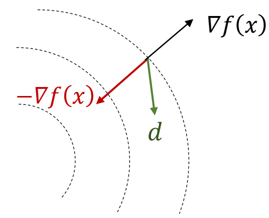
The rate of increase of f in the direction d at the point x

$$Df(x,d) = \langle \nabla f(x), d \rangle, \qquad \|d\| = 1$$

By the Cauchy-Schwarz inequality,

$$\langle \nabla f(x), d \rangle \le \|\nabla f(x)\| \|d\| = \|\nabla f(x)\|$$

$$d = \frac{\nabla f(x)}{\|\nabla f(x)\|} \qquad \left\langle \nabla f(x), \frac{\nabla f(x)}{\|\nabla f(x)\|} \right\rangle = \|\nabla f(x)\|$$



$\nabla f(x)$ is the direction of maximum rate of increase of f at x

The direction of negative gradient is a good direction to search for a minimizer

获得一个使函数值更优的点

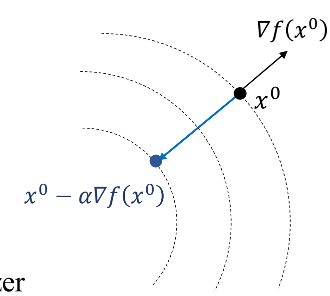
Consider $x^0 - \alpha \nabla f(x^0)$, $\quad \alpha > 0$

By Taylor's theorem

$$f\left(x^0 - \alpha \nabla f(x^0)\right) = f(x^0) - \alpha \|\nabla f(x^0)\|^2 + o(\alpha)$$

$\nabla f(x^0) \ne 0$

$$f\left(x^0 - \alpha \nabla f(x^0)\right) < f(x^0)$$



$x^0 - \alpha \nabla f(x^0)$ is an improvement over the point x^0 for a minimizer

梯度下降法 Gradient Descent Algorithm

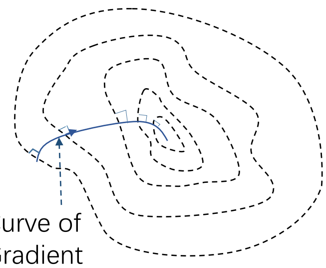
Start at x^k and move by an amount $-\alpha_k \nabla f(x^k)$

the *step size,* $\alpha_k > 0$

$$x^{k+1} = x^k - \alpha_k \nabla f(x^k) \qquad f(x^{k+1}) \le f(x^k)$$

$$\nabla f(x^k) \to 0$$

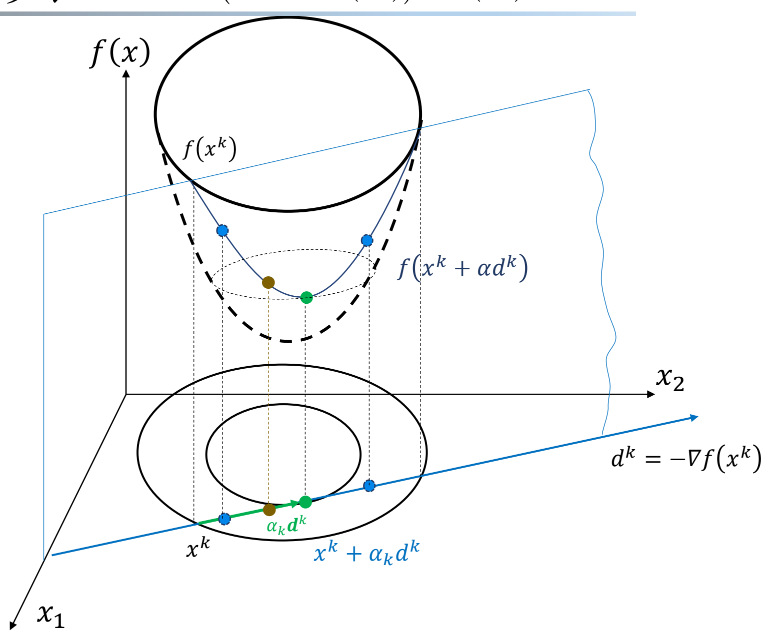
The gradient varies as the search proceeds, tending to zero as we approach the minimizer



梯度下降法 $d^k = -\nabla f(x^k)$为搜索（下降）方向

$$x^{k+1} = x^k - \alpha \nabla f(x^k) \qquad f(x^{k+1}) \le f(x^k)$$

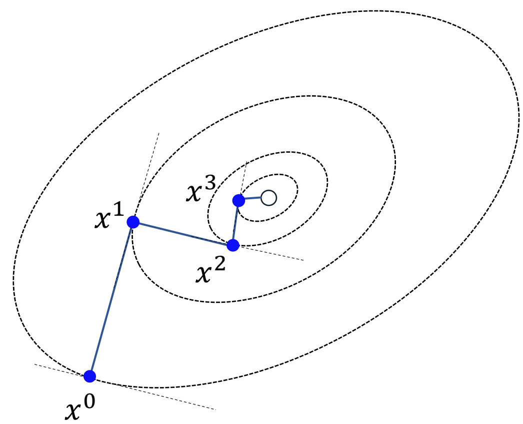
如何确定步长α？ $f\left(x^k - \alpha \nabla f\left(x^k\right)\right) < f\left(x^k\right)$



• 最速下降法 Steepest Descent

At each step,
starting from the point x^k
we conduct a line search in the direction $-\nabla f(x^k)$
until a minimizer, x^{k+1}, is found

$$\alpha_k = \underset{\alpha \ge 0}{\operatorname{argmin}} f\left(x^k - \alpha \nabla f(x^k)\right)$$



注：与‘Convex Optimization’，Boyd中的概念不同

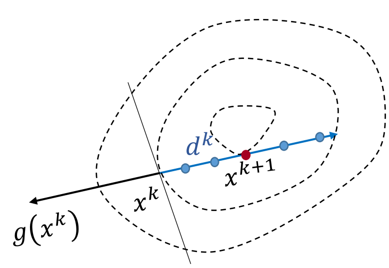
最速下降法的原理

目标函数$f:\mathcal{R}^n \to \mathcal{R}$（$n>1$），$x^{k+1} = x^k + \alpha_k d^k$，$d^k = -\nabla f(x^k)$，$g(x^k) = \nabla f(x^k)$

$$\alpha_k = \underset{\alpha \ge 0}{\operatorname{argmin}} f\left(x^k - \alpha \nabla f(x^k)\right)$$

$$f\left(x^k - \alpha_k \nabla f(x^k)\right) \le f\left(x^k - \alpha \nabla f(x^k)\right)$$

下降量$f(x^k) - f(x^{k+1})$最大



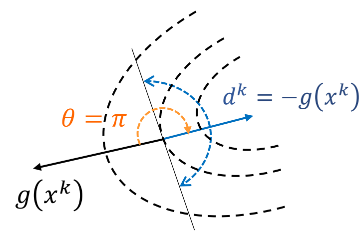
最速下降法的特点

① 若x^k不是极小点，则f在点x^k处的最速下降方向总是下降方向

证：x^k不是极小点→ $g(x^k) \neq 0$

$$g(x^k)^T d^k = -g(x^k)^T g(x^k) = -\|g(x^k)\|^2 < 0$$

$$d^k = -g(x^k)\text{是下降方向}$$



② 如果每次迭代时都用精确搜索法得到最佳步长作为搜索步长，则迭代过程中相邻的最速下降方向是正交的

解释

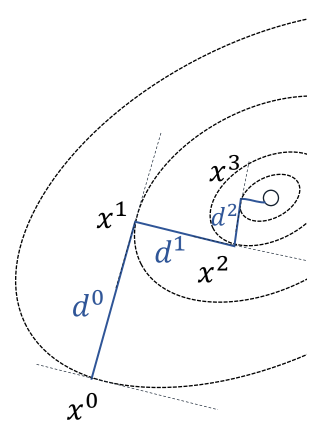
- d^{k+1}与d^k正交

  相邻的迭代点的搜索方向正交

- g^{k+1}与d^k正交

  当前点的梯度与上一点的搜索方向正交

  相邻的迭代点的梯度正交



采用精确最佳步长，搜索方向与梯度方向垂直

② 如果每次迭代时都用精确搜索法得到最佳步长作为搜索步长，则迭代过程中相邻的最速下降方向是正交的

证：$x^{k+1} = x^k + \alpha d^k$

对最佳步长有，$\left.\frac{df(x^k+\alpha d^k)}{d\alpha}\right|_{\alpha=\alpha^*} = 0$

$$\frac{df(x^k+\alpha d^k)}{d\alpha} = D_x\left(f(x^k+\alpha d^k)\right)\cdot D_\alpha(x^k+\alpha d^k) = g(x^k+\alpha d^k)^T d^k$$

$$\left.\frac{df(x^k+\alpha d^k)}{d\alpha}\right|_{\alpha=\alpha^*} = g(x^k+\alpha^* d^k)^T d^k = g^{k+1\,T} d^k = -d^{k+1\,T} d^k = 0$$

d^{k+1}与d^k正交　　g^{k+1}与d^k正交

最速下降法的线性收敛性

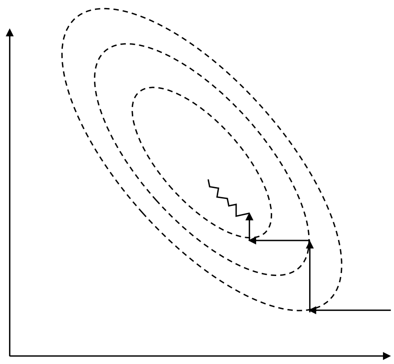
设函数$f(x)$二阶可导，x^*是局部极小点，x^*处的Hesse矩阵$H(x^*) \succ 0$，其最小特征值和最大特征值分别为$\lambda_{\min}$和$\lambda_{\max}$. 如果最速下降法产生的迭代点序列$\{x^k\}$充分接近x^*，则目标函数值的序列$\{f(x^k)\}$以不大于β的收敛比线性收敛于$f(x^*)$

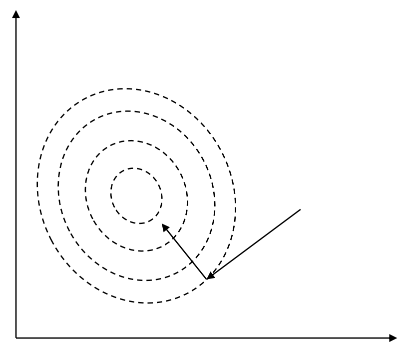
$$\frac{f(x^{k+1}) - f(x^*)}{f(x^k) - f(x^*)} \le \beta \qquad \beta = \frac{(r-1)^2}{(r+1)^2} \qquad r = \frac{\lambda_{\max}}{\lambda_{\min}}$$ 是矩阵$H(x^*)$的条件数

当矩阵$H(x^*)$特征值相差不大时，$r \to 1$　$\beta \to 0$，收敛速度非常快

当矩阵$H(x^*)$特征值相差大时，　$r \to 0$　$\beta \to 1$，收敛速度非常慢

$f(\boldsymbol{x})$是二次多项式凸函数，Hesse矩阵的特征值λ_1和λ_2，$0 < \lambda_1 \leq \lambda_2$，
椭圆长、短轴$1/\sqrt{\lambda_1}$、$1/\sqrt{\lambda_2}$
两个特征值相差大时，　椭圆趋于扁平，最速下降法收敛较慢
两个特征值相差不大时，椭圆趋于正圆，最速下降法收敛较快





最速下降法的收敛速度与$H(\boldsymbol{x}^*)$的关系

最速下降法的计算步骤

步骤1：已知目标函数$f(x)$，初始点x^0，精度要求tol。$k = 0$

步骤2：计算$d^k = -g(x^k)$；

若$\|g(x^k)\| < tol$，则终止迭代，输出x^k及$f(x^k)$

步骤3：从x^k出发，沿方向d^k进行精确一维搜索得到最佳步长α_k

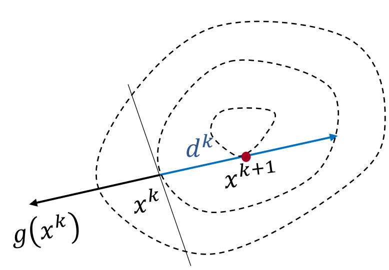
$$\alpha_k = \underset{\alpha \geq 0}{\operatorname{argmin}} f\left(x^k - \alpha \nabla f(x^k)\right)$$

步骤4：计算新点$x^{k+1} = x^k + \alpha_k d^k$及$f(x^{k+1})$

步骤5：若$\|x^{k+1} - x^k\| < tol[\min\{1, \|\boldsymbol{x}^k\|\}]$，则终止迭代，

输出x^{k+1}及$f(x^{k+1})$；

否则，$k = k + 1$，转步骤2



1.引言

2.梯度下降法

- 最速下降法
- 一般梯度下降法

3.牛顿法

梯度下降法的计算步骤

步骤1：已知目标函数$f(x)$，初始点x^0，精度要求tol。$k=0$

步骤2：计算$d^k=-g(x^k)$；

若$\|g(x^k)\|<tol$，则终止迭代，输出x^k及$f(x^k)$

步骤3：从x^k出发，沿方向d^k进行非精确一维搜索得到可接受步长α_k

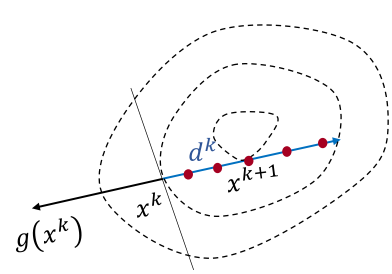
一般地，$\alpha_k \neq \operatorname*{argmin}_{\alpha\geq 0} f\left(x^k-\alpha\nabla f(x^k)\right)$

步骤4：计算新点$x^{k+1}=x^k+\alpha_k d^k$及$f(x^{k+1})$

步骤5：若$\|x^{k+1}-x^k\|<tol[\min\{1,\|\boldsymbol{x}^k\|\}]$，则终止迭代，

输出x^{k+1}及$f(x^{k+1})$；

否则，$k=k+1$，转步骤2



Example 8.1

We use the method of steepest descent to find the minimizer of

$$f(\boldsymbol{x}) = (x_1 - 4)^4 + (x_2 - 3)^2 + 4(x_3 + 5)^4$$

The initial point is $\boldsymbol{x}^0 = [4 \quad 2 \quad -1]^T$. We perform three iterations.

SOLUTION

$$\nabla f(\boldsymbol{x}) = [4(x_1 - 4)^3 \quad 2(x_2 - 3) \quad 16(x_3 + 5)^3]^T$$

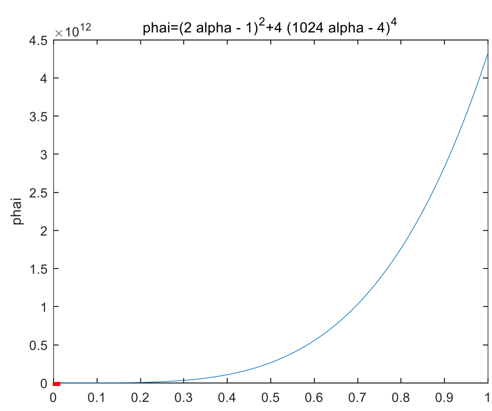
$$\nabla f(\boldsymbol{x}^0) = [0 \quad -2 \quad 1024]^T$$

$$\begin{aligned} \boldsymbol{x}^0 - \alpha \nabla f(\boldsymbol{x}^0) &= [4 \quad 2 \quad -1]^T - \alpha[0 \quad -2 \quad 1024]^T \\ &= [4 \quad 2 + 2\alpha \quad -1 - 1024\alpha]^T \end{aligned}$$

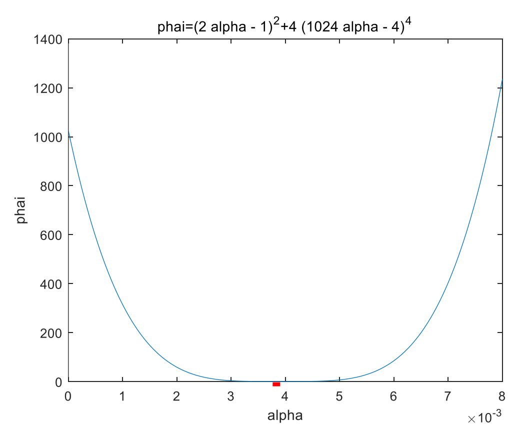
$$\begin{aligned} \phi_0(\alpha) &= f\left(\boldsymbol{x}^0 - \alpha \nabla f(\boldsymbol{x}^0)\right) \\ &= (4-4)^4 + (2 + 2\alpha - 3)^2 + 4(-1 - 1024\alpha + 5)^4 \\ &= (2\alpha - 1)^2 + 4(1024\alpha - 4)^4 \end{aligned}$$

To compute $\boldsymbol{x}^1$, $\quad \alpha_0 = \underset{\alpha > 0}{\operatorname{argmin}} \, \phi_0(\alpha)$

观察曲线 $\phi_0(\alpha) = f\left(\boldsymbol{x}^0 - \alpha \nabla f(\boldsymbol{x}^0)\right) = (2\alpha - 1)^2 + 4(1024\alpha - 4)^4$

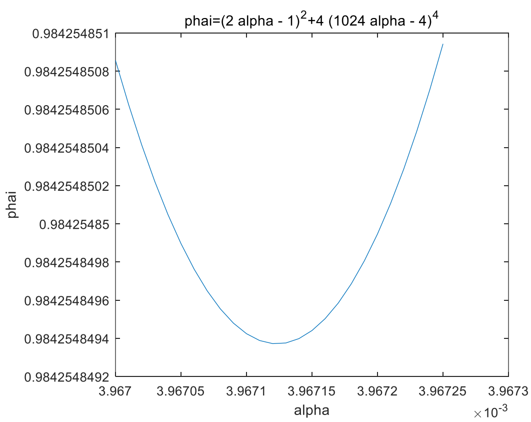


[0 1]



[0 0.008]

$$\phi_0(\alpha) = f\left(\boldsymbol{x}^0 - \alpha \nabla f(\boldsymbol{x}^0)\right) = (2\alpha - 1)^2 + 4(1024\alpha - 4)^4$$



$10^{-3}[3.967 \quad 3.9675]$

用割线法求最佳步长

$$\alpha_{i+1} = \alpha_i - \frac{\alpha_i - \alpha_{i-1}}{\phi_k'(\alpha_i) - \phi_k'(\alpha_{i-1})}\phi_k'(\alpha_i)$$

$$\boxed{\phi_k'(\alpha) = \ -{g^k}^T \nabla f\left(x^k - \alpha g^k\right)}$$

$$\phi_k(\alpha) \triangleq f\left(x^k - \alpha g^k\right) \qquad g^k = \nabla f\left(x^k\right)$$

$$f(x) = (x_1 - 4)^4 + (x_2 - 3)^2 + 4(x_3 + 5)^4$$

$$\nabla f(x) = [4(x_1 - 4)^3 \quad 2(x_2 - 3) \quad 16(x_3 + 5)^3]^T$$

$$x^0 = [4 \quad 2 \quad -1]^T$$

$$g^0 = [0 \quad -2 \quad 1024]^T$$

$$x^0 - \alpha g^0 = [4 \quad 2 \quad -1]^T - \alpha[0 \quad -2 \quad 1024]^T$$
$$= [4 \quad 2 + 2\alpha \quad -1 - 1024\alpha]^T$$

$$\nabla f(x^0 - \alpha g^0) = [0 \quad 2(2\alpha - 1) \quad 16(-1024\alpha + 4)^3]^T$$

$$\phi_0'(\alpha) = -{g^0}^T \nabla f(x^0 - \alpha g^0)$$
$$= 4(2\alpha - 1) + 16384(1024\alpha - 4)^3$$

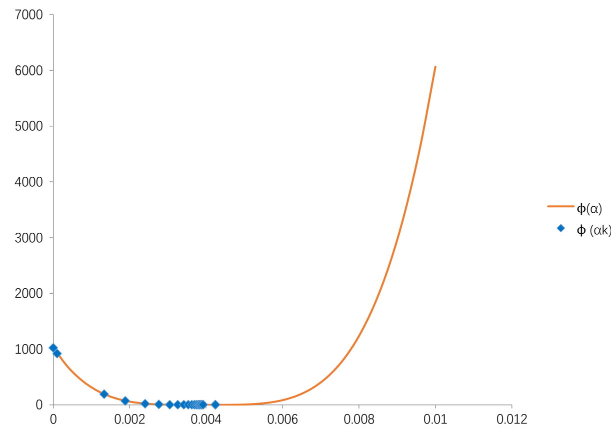
用割线法求最佳步长

$$\phi_0'(\alpha) = 4(2\alpha - 1) + 16384(1024\alpha - 4)^3$$

$$\alpha_{i+1} = \alpha_i - \frac{\alpha_i - \alpha_{i-1}}{\phi_0'(\alpha_i) - \phi_0'(\alpha_{i-1})}\phi_0'(\alpha_i)$$

$$\alpha_0 = 3.967 \times 10^{-3}$$

$$\begin{aligned} x^1 &= x^0 - \alpha_0 \nabla f(x^0) \\ &= [4 \quad 2 + 2\alpha_0 \quad -1 - 1024\alpha_0]^T \\ &= [4.000 \quad 2.008 \quad -5.062]^T \end{aligned}$$



To compute x^2, $\nabla f(x^1) = [0.000 \quad -1.984 \quad -0.003875]^T$

$$\begin{aligned} x^1 - \alpha \nabla f(x^1) &= [4.000 \quad 2.008 \quad -5.062]^T - \alpha[0 \quad -1.984 \quad -0.003875]^T \\ &= [4 \quad 2.008 + 1.984\alpha \quad -5.062 + 0.003875\alpha]^T \end{aligned}$$

$$\begin{aligned} \phi_1(\alpha) &= f\left(x^1 - \alpha \nabla f(x^1)\right) \\ &= (0 + (2.008 + 1.984\alpha - 3)^2 + 4(-5.062 + 0.003875\alpha + 5)^4) \\ &= (1.984\alpha - 0.992)^2 + 4(0.003875\alpha - 0.062)^4 \end{aligned}$$

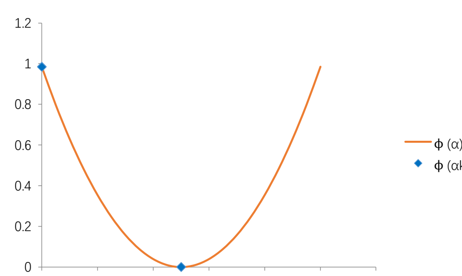
$$\alpha_1 = \underset{\alpha \geq 0}{\operatorname{argmin}}\, \phi_1(\alpha)$$

$$\phi_1'(\alpha) = 3.968(1.984\alpha - 0.992) + 0.062(0.003875\alpha - 0.062)^3$$

$$\alpha_{i+1} = \alpha_i - \frac{\alpha_i - \alpha_{i-1}}{\phi_1'(\alpha_i) - \phi_1'(\alpha_{i-1})}\phi_1'(\alpha_i)$$

$$\alpha_1 = 0.5000$$

$$x^2 = x^1 - \alpha_1 \nabla f(x^1) = [4.000 \quad 3.000 \quad -5.060]^T$$



To compute x^3, $\nabla f(x^2) = [0 \quad 0 \quad -0.00346]^T$

$$x^2 - \alpha \nabla f(x^2) = [4.000 \quad 3 \quad -5.062]^T - \alpha[0 \quad 0 \quad -0.00346]^T$$
$$= [4 \quad 3 \quad -5.060 + 0.00346\alpha]^T$$

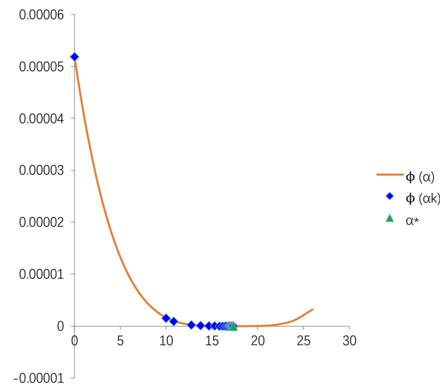
$$\phi_2(\alpha) = 4(0.00346\alpha - 0.060)^4$$

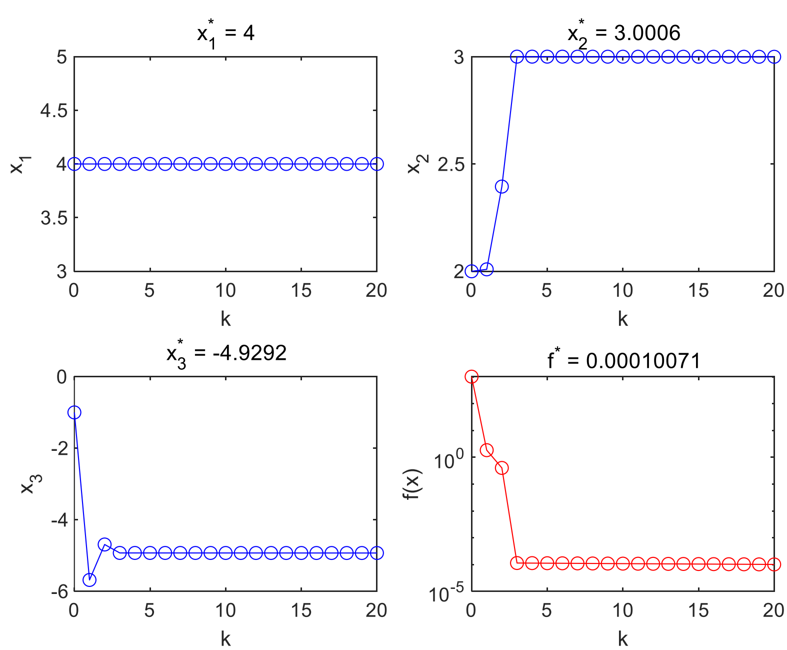
$$\phi_2'(\alpha) = 0.0554(0.00346\alpha - 0.060)^3$$

$$\alpha_{i+1} = \alpha_i - \frac{\alpha_i - \alpha_{i-1}}{\phi_2'(\alpha_i) - \phi_2'(\alpha_{i-1})} \phi_2'(\alpha_i)$$

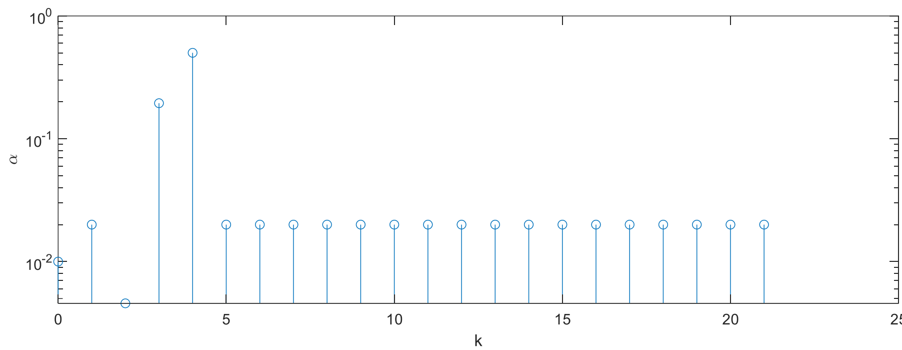
$$\alpha_2 = 16.29$$

$$x^3 = x^2 - \alpha_2 \nabla f(x^2)$$
$$= [4.000 \quad 3.000 \quad -5.0036]^T$$






Example08_01_Chong.m



编程解释

主函数

- ✓ 赋初值
- ✓ 调用子函数 1： 求步长和新点[a,xn]=SecantMethod(@dpfun,xk,dk,a00,a0,epsi1)

  子函数 2： 函数值 ‘dpfun’

  子函数 3： 梯度

  子函数 5： 画图

子函数1 “割线法” [a,xn]=SecantMethod(fun,xk,dk,a00,a0,epsi1)

调用子函数4 求 导数

子函数2 函数值 $f(x)$

子函数3 梯度 $g(x) = \nabla f(x)$

子函数4 导数 $\phi'(\alpha) = {d^k}^T \nabla f(x^k + \alpha d^k)$

$\phi(\alpha) = f(x^k + \alpha d^k)$

子函数5 画图 FigureExample08_1(k,ar,xr,fr)

主函数
- ✓ 赋初值
- ✓ 调用子函数1：求步长和新点

子函数2：　函数值
子函数3：　梯度
子函数5：画图

```
function Example08_01_Chong
close all;clear all;clc;
x0=[4;2;-1];
kmax=21;
epsi=1.e-8;   epsi1=1.e-8;
a00=1.e-2;    a0=a00+1.e-2;
xk=x0;xr=zeros(length(x0),kmax);fr=zeros(kmax,1);
k=1;
xr = zeros(length(x0),kmax); xr(:,k)=xk;
fr = zeros(kmax,1);        fr(k)= fun(xk);
ar = ones(kmax+1,1);       ar(1)=a00;ar(2)=a0;
gk = gfun(xk);

while norm(gk,2) > epsi && k < kmax
  dk = -gk;
  [ar(k+2),xn] = SecantMethod(@dpfun,xk,dk,a00,a0,epsi1);
  k=k+1;
  xk=xn;      gk   = gfun(xk);
  xr(:,k)=xk;  fr(k) = fun(xk);
end
fr=fr(1:k);   ar=ar(1:k+1);   xr=xr(:,1:k);
FigureExample08_1(k,ar,xr,fr)
end
```

α_{-1}, α_0

记录迭代点
记录迭代函数值
记录步长

按迭代次数截断

子函数1　“割线法”　[a,xn]=SecantMethod(fun,xk,dk,a00,a0,epsi1)
调用子函数4 求导数

fun = $\phi'(\alpha)$

$$\alpha_{i+1} = \alpha_i - \frac{\alpha_i - \alpha_{i-1}}{\phi_k'(\alpha_i) - \phi_k'(\alpha_{i-1})}\phi_k'(\alpha_i)$$

```
function [a,xn] = SecantMethod(fun,xk,dk,a00,a0,epsi1)
% dphi at a00,a0
kmax=10;
al=a00;ac=a0;k=1;
dphil = fun(al,xk,dk);
dphic = fun(ac,xk,dk);
while abs(dphic) > epsi1 && abs(dphic-dphil) > 1.e-3 && k < kmax
  an = ac-(ac-al)*dphic/(dphic-dphil);
  k = k+1;
  al = ac; dphil = dphic;
  ac = an; dphic = fun(ac,xk,dk);
end
a = ac; xn = xk+a*dk;
end
```

子函数 2:　　函数值
子函数 3:　　梯度
子函数5:　画图

```
function f = fun(x)
f = (x(1)-4).^4+(x(2)-3).^2+4*(x(3)+5).^4;
end
```

```
function g = gfun(x)
g = [4*(x(1)-4).^3;2*(x(2)-3);16*(x(3)+5).^3];
end
```

子函数4　导数　$\phi'(\alpha) = {d^k}^T \nabla f\left(x^k + \alpha d^k\right)$

$$\phi(\alpha) = f\left(x^k + \alpha d^k\right)$$

```
function dphi = dpfun(a,xk,dk)
x = xk + a*dk;
dphi = dk'*[4*(x(1)-4).^3;2*(x(2)-3);16*(x(3)+5).^3];
end
```

子函数 2:　　函数值
子函数 3:　　梯度
子函数5:　画图

```
function FigureExample08_1(k,ar,xr,fr)
%----figure step-size----------------
figure(1)
stem(0:k,ar)
xlabel('k');ylabel('\alpha');
%---- k-x k-f ---------------
figure(2)
subplot(2,2,1)
plot(0:k-1,xr(1,1:k),'-ob')
xlabel('k');ylabel('x_1');
str1=num2str(xr(1,k));   str=['x^*_1 = ',str1]; title(str)
subplot(2,2,2)
plot(0:k-1,xr(2,1:k),'-ob')
xlabel('k');ylabel('x_2');
str2=num2str(xr(2,k));   str=['x^*_2 = ',str2];  title(str)
subplot(2,2,3)
plot(0:k-1,xr(3,1:k),'-ob')
xlabel('k');ylabel('x_3');
str3=num2str(xr(3,k));   str=['x^*_3 = ',str3];  title(str)
subplot(2,2,4)
semilogy(0:k-1,fr,'-or')
xlabel('k');ylabel('f(x)');
str0=num2str(fr(end));   str=['f^* = ',str0];   title(str)
end
```

梯度下降法的MATLAB程序

Steepest_Descent.m

增加非零梯度的要求

norm(x_next-x_current)>tolerance&&norm(g_current)>g_tolerance

Wolfe_search.m

例

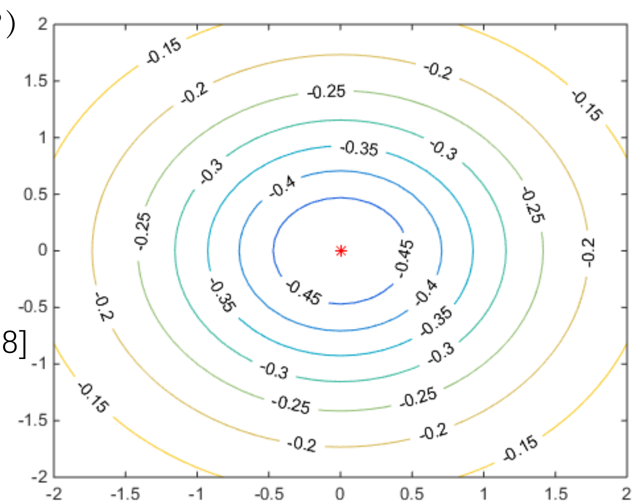
例5.1 用最速下降法求解多维无约束优化问题

（取初始点$\boldsymbol{x}^0=(2,2), tol=1\times10^{-6}$）

$$\min f(\boldsymbol{x})=-\frac{1}{x_1^2+x_2^2+2}$$

example_5_1_CH05.m

x_optimal =1.0e-17 *[-0.4018   -0.4018]
f_optimal = -0.5000
k = 4



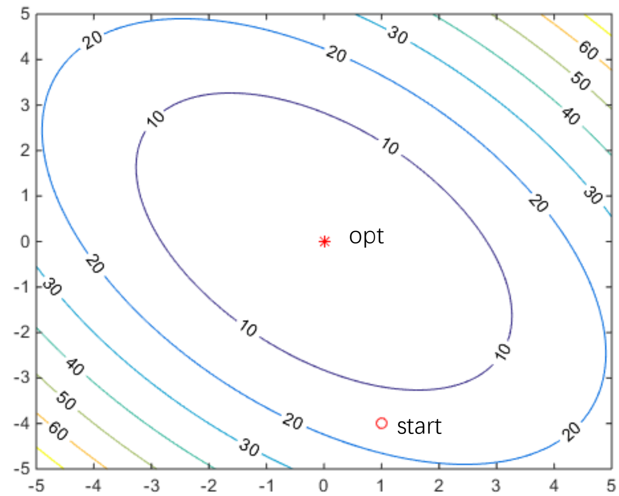
例5.2 用最速下降法求解多维无约束优化问题

（取初始点$\boldsymbol{x}^0 = (1,-4), tol = 1 \times 10^{-6}$）

$$\min f(\boldsymbol{x}) = x_1^2 + x_2^2 + x_1 x_2 + 2$$

example_5_2_CH05.m

x_optimal = 1.0e-07 *[0.4437   -0.3911]
f_optimal = 2.0000
k = 12



例5.3 用最速下降法求解多维无约束优化问题

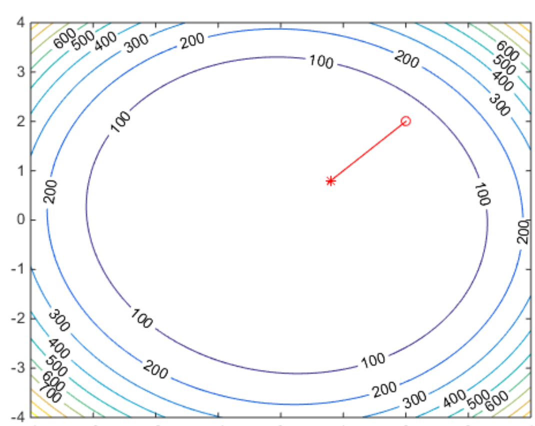
（取初始点$\boldsymbol{x}^0 = (2,2), tol = 1 \times 10^{-6}$）

$$\min f(\boldsymbol{x}) = (x_1^2 + x_2^2 - 1)^2 + (x_1 + x_2 - 2)^2$$

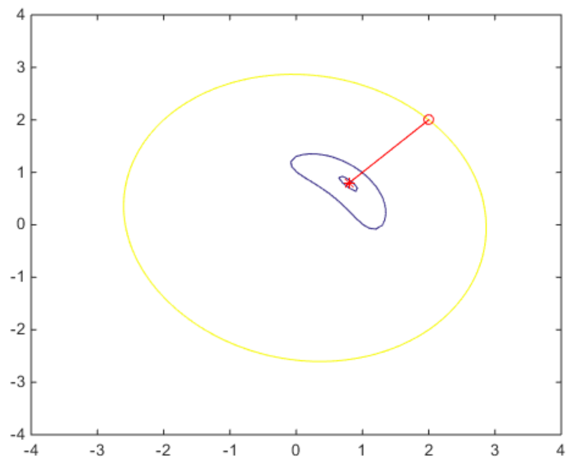
example_5_3_CH05.m

x_optimal =[0.7937   0.7937]
f_optimal = 0.2378
k = 4

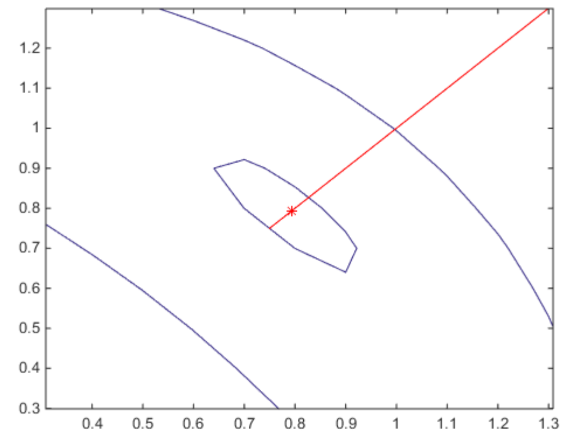
k	x^(k)		f^(k)
0	2.0000	2.0000	53.0000
1	0.9974	0.9974	0.9791
2	0.7487	0.7487	0.2673
3	0.7944	0.7944	0.2378
4	0.7937	0.7937	0.2378
5	0.7937	0.7937	0.2378



迭代点所在的等值线



局部放大



实用性强

2. 注意:

当$\|d_current\| = 0$ 或者 在给定最大搜索次数内找不到可接受步长时,

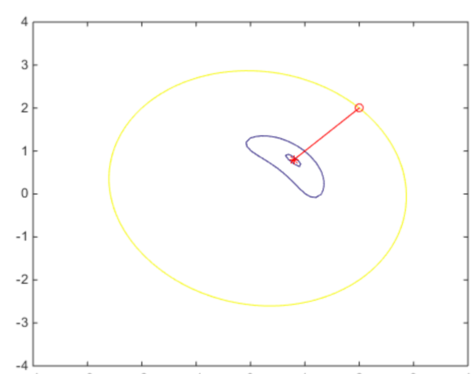
可接受步长 = 0, x_next = x_current, f_next = f_current

x_optimal =[0.7937   0.7937]
f_optimal = 0.2378
k = 4

example_5_3___CH05.m

Steepest__Descent.m

Wolfe__Search.m



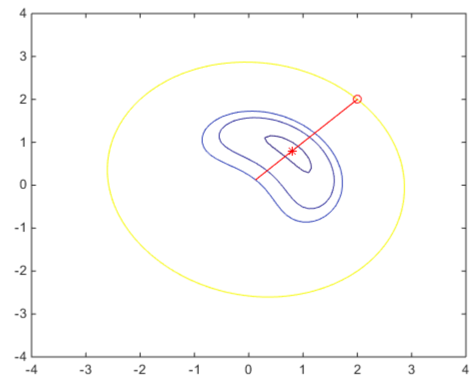
3.采用Armijo搜索法计算可接受步长

x_optimal =[0.7937 0.7937]
f_optimal = 0.2378
k = 8

example_5_3_CH05_Armijo.m

Steepest_Descent_Armijo.m

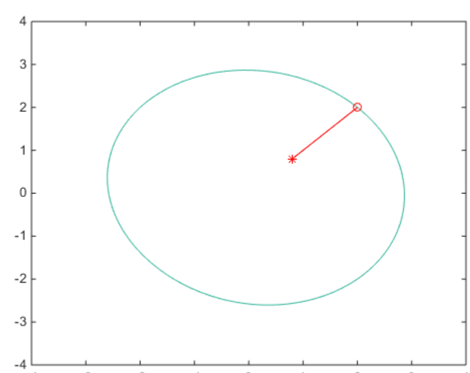
armijo_mk.m



4.采用陆吾生教授的Fletcher搜索法(Wolfe准则的实现)计算可接受步长
PracticalOptimizationAlgorithmsandEngineeringApplications-A.Antoniou,W.Lu-Springer-2007 P126
inex_lsearch_01.m
Steepest_Descent_InexactSearch.m
example_5_3_CH05_InexactSearch.m
Testdata.txt

算法稳定性好

k	x^(k)		f^(k)
0	2.000000	2.000000	53.000000
1	0.794511	0.794511	0.237807
2	0.793284	0.793284	0.237799
3	0.793913	0.793913	0.237798
4	0.793592	0.793592	0.237797
5	0.793756	0.793756	0.237797
6	0.793672	0.793672	0.237797
7	0.793715	0.793715	0.237797
8	0.793693	0.793693	0.237797
9	0.793704	0.793704	0.237797
10	0.793704	0.793704	0.237797



- The Method of Steepest Descent for a positive definite quadratic function

Quadratic function $\boxed{f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T Q\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}, \qquad Q \succ 0}$ $Q \in \mathcal{R}^{n\times n}$, $\boldsymbol{b} \in \mathcal{R}^n$, and $\boldsymbol{x} \in \mathcal{R}^n$

$$\boldsymbol{g}^k = \nabla f(\boldsymbol{x}^k)$$
$$\nabla f(\boldsymbol{x}) = Q\boldsymbol{x} - \boldsymbol{b} \rightarrow \boldsymbol{g}^k = Q\boldsymbol{x}^k - \boldsymbol{b}$$

$$\begin{aligned}\nabla f(\boldsymbol{x}^k - \alpha\boldsymbol{g}^k) &= Q(\boldsymbol{x}^k - \alpha\boldsymbol{g}^k) - \boldsymbol{b} \\ &= Q\boldsymbol{x}^k - \alpha Q\boldsymbol{g}^k - \boldsymbol{b} \\ &= (Q\boldsymbol{x}^k - \boldsymbol{b}) - \alpha Q\boldsymbol{g}^k \\ &= \boldsymbol{g}^k - \alpha Q\boldsymbol{g}^k\end{aligned}$$

$$\phi_k(\alpha) = f(\boldsymbol{x}^k - \alpha\boldsymbol{g}^k)$$

$$\alpha_k = \arg\min_{\alpha\geq 0} f(\boldsymbol{x}^k - \alpha\boldsymbol{g}^k)$$

$$\begin{aligned}0 = \phi'_k(\alpha) &= Df(\boldsymbol{x}^k - \alpha\boldsymbol{g}^k)D(\boldsymbol{x}^k - \alpha\boldsymbol{g}^k) \\ &= \left(\nabla f(\boldsymbol{x}^k - \alpha\boldsymbol{g}^k)\right)^T(-\boldsymbol{g}^k) \\ &= (\boldsymbol{g}^k - \alpha Q\boldsymbol{g}^k)^T(-\boldsymbol{g}^k) \\ &= -\boldsymbol{g}^{k\,T}\boldsymbol{g}^k + \alpha\boldsymbol{g}^{k\,T}Q\boldsymbol{g}^k\end{aligned}$$

$$\alpha_k = \frac{\boldsymbol{g}^{k^T}\boldsymbol{g}^k}{\boldsymbol{g}^{k^T}Q\boldsymbol{g}^k}$$

$$\boxed{\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \frac{\boldsymbol{g}^{k^T}\boldsymbol{g}^k}{\boldsymbol{g}^{k^T}Q\boldsymbol{g}^k}\boldsymbol{g}^k}$$

The Method of Steepest Descent for a positive definite quadratic function

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T Q\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}, \qquad Q \succ 0$$

$$\boldsymbol{g}^k = Q\boldsymbol{x}^k - \boldsymbol{b}$$

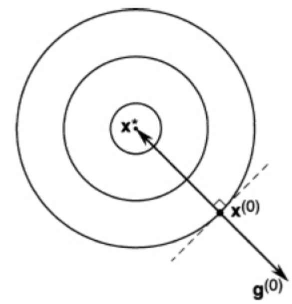
$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \frac{\boldsymbol{g}^{k^T}\boldsymbol{g}^k}{\boldsymbol{g}^{k^T}Q\boldsymbol{g}^k}\boldsymbol{g}^k$$

Example 8.2

Let $f(\boldsymbol{x}) = x_1^2 + x_2^2$

Starting from an arbitrary initial point $\boldsymbol{x}^{(\mathbf{0})} \in \mathcal{R}^2$,

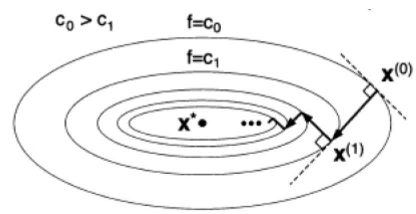
we arrive at the solution $\boldsymbol{x}^* = \mathbf{0} \in \mathcal{R}^2$ in only one step.



Let $f(\boldsymbol{x}) = \frac{1}{5}x_1^2 + x_2^2$

The method of steepest descent shuffles ineffectively back and forth when searching for the minimizer in a narrow valley.

This example illustrates a major drawback in the steepest descent method.



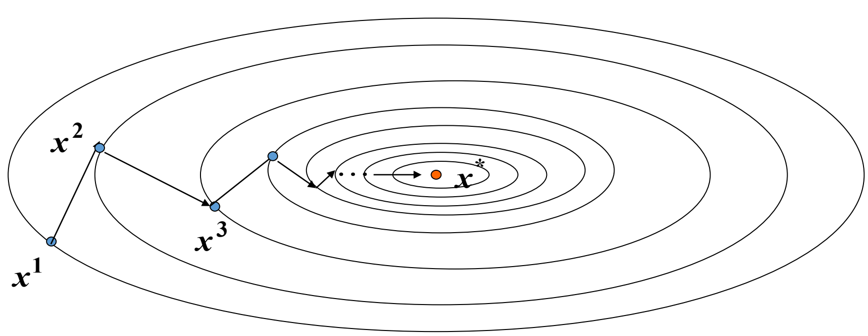
zigzag

A Major Drawback

Near the optimum $\boldsymbol{x}^*$, any function can be approximated by some quadratic function. So, we can simply draw a contour surface – circles.

At the beginning, the derivative is large, when approaching to the minimum, the gradient is changed to flat slowly along its descent zigzag.

The optimum of $f(\boldsymbol{x})$ may not be attained in a finite distance.



Fixed-step-size Gradient Algorithm

Consider $\alpha_k = \alpha$ for all k in a gradient method

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha \boldsymbol{g}^k$$

We refer to the algorithm above as a *fixed-step-size* gradient algorithm.

For the fixed-step-size gradient algorithm, $\boldsymbol{x}^k \to \boldsymbol{x}^*$ for any $\boldsymbol{x}^0$

if and only if $0 < \alpha < \frac{2}{\lambda_{\max}(Q)}$.

Example 8.4

Let the function f be given by $f(x) = x^{\mathrm{T}} \begin{bmatrix} 4 & 2\sqrt{2} \\ 0 & 5 \end{bmatrix} x + x^{\mathrm{T}} \begin{bmatrix} 3 \\ 6 \end{bmatrix} + 24$

We wish to find the minimizer of f using a fixed-step-size gradient algorithm where $\alpha \in \mathcal{R}$ is a fixed step size.

$$Q = A + A^{\mathrm{T}} = \begin{bmatrix} 4 & 2\sqrt{2} \\ 0 & 5 \end{bmatrix} + \begin{bmatrix} 4 & 0 \\ 2\sqrt{2} & 5 \end{bmatrix} = \begin{bmatrix} 8 & 2\sqrt{2} \\ 2\sqrt{2} & 10 \end{bmatrix} \qquad b = \begin{bmatrix} -3 \\ -6 \end{bmatrix}$$

$$f(x) = \frac{1}{2} x^T Q x - x^T b + 24$$

$$\lambda_{max} = \max\{6,12\} = 12$$

$$0 < \alpha < \frac{2}{12} = \frac{1}{6}$$

$$x^{k+1} = x^k - \alpha g^k,$$
$$g^k = Q x^k - b$$

1.引言

2.梯度下降法

3.牛顿法

- 牛顿法
- 阻尼牛顿法
- Levenberg-Marquardt牛顿法
- 高斯牛顿法

牛顿法基本思想

$k=0$

Construct a quadratic function $q: \mathcal{R}^n \to \mathcal{R}, q \in C^2$

$$q(x_k)=f(x_k), \quad \nabla q(x_k)=\nabla f(x_k), \quad \nabla^2 q(x_k)=\nabla^2 f(x_k)$$

$$\min f(x) \Leftarrow \min q(x), \quad \text{near} x_k$$

$$x^* = \operatorname{argmin} q(x), x_{k+1}=x^*$$

$k=k+1$

$\|x_0-x^*\|<\delta, \delta>0$ small number

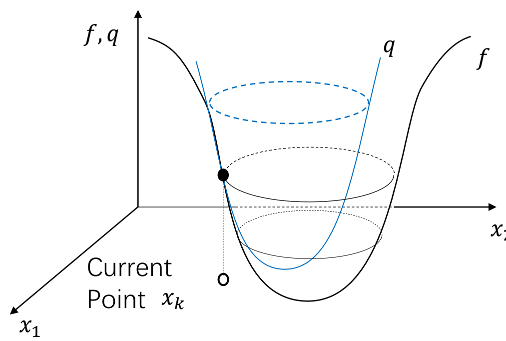
Quadratic Approximation $q(\boldsymbol{x})$

By the Taylor series

$$q(x) \triangleq f(x_k) + (x - x_k)^T g_k + \frac{1}{2}(x - x_k)^T H_k (x - x_k) \approx f(x)$$

$$g_k = \nabla f(x_k)$$

$$H_k = \nabla^2 f(x_k)$$



$\|x_0 - x^*\| < \delta$, $\delta > 0$ small number

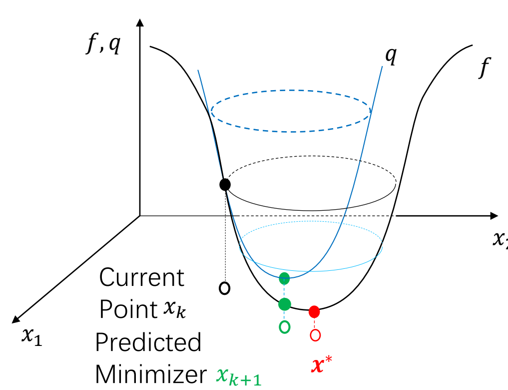
Recursive Formula of Newton's method

$$q(x) \triangleq f(x_k) + (x - x_k)^T g_k + \frac{1}{2}(x - x_k)^T H_k (x - x_k)$$

By FONC, $\nabla q(x) = g_k + H_k(x - x_k) = 0$

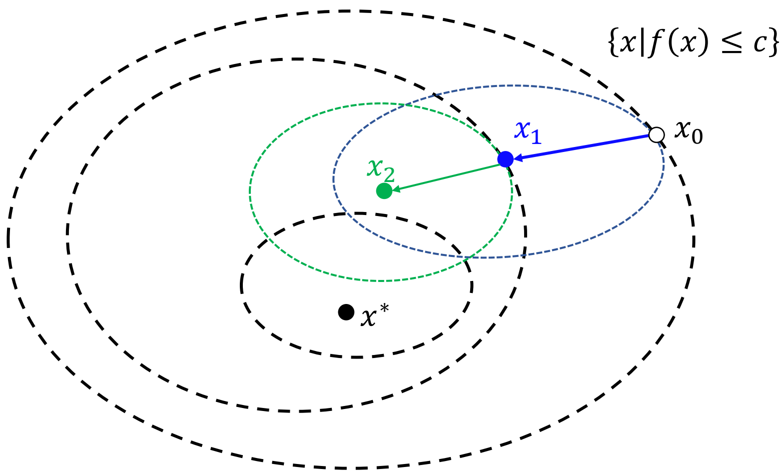
If $H_k \succ 0$, q achieves a minimum at

$$x_{k+1} = x_k - H_k^{-1} g_k$$



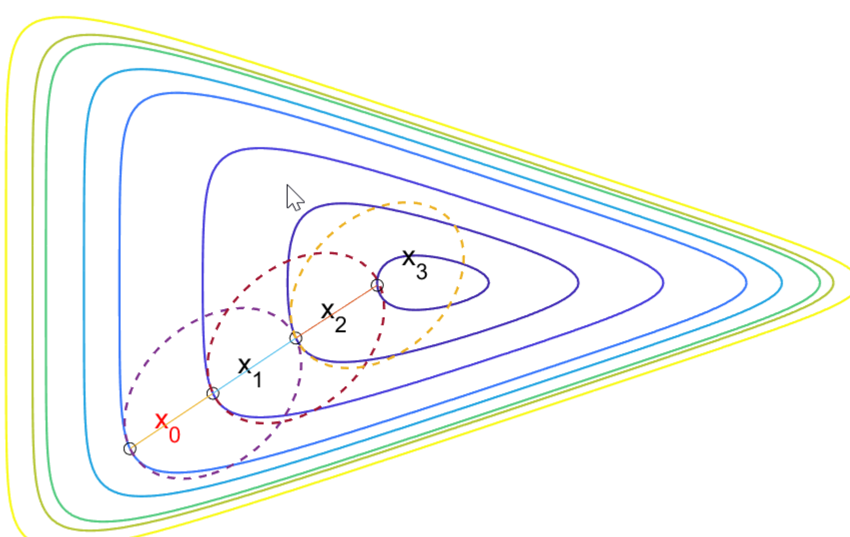
牛顿法迭代的几何直观

取 x^0 与 x^* 比较远，为图示清楚起见

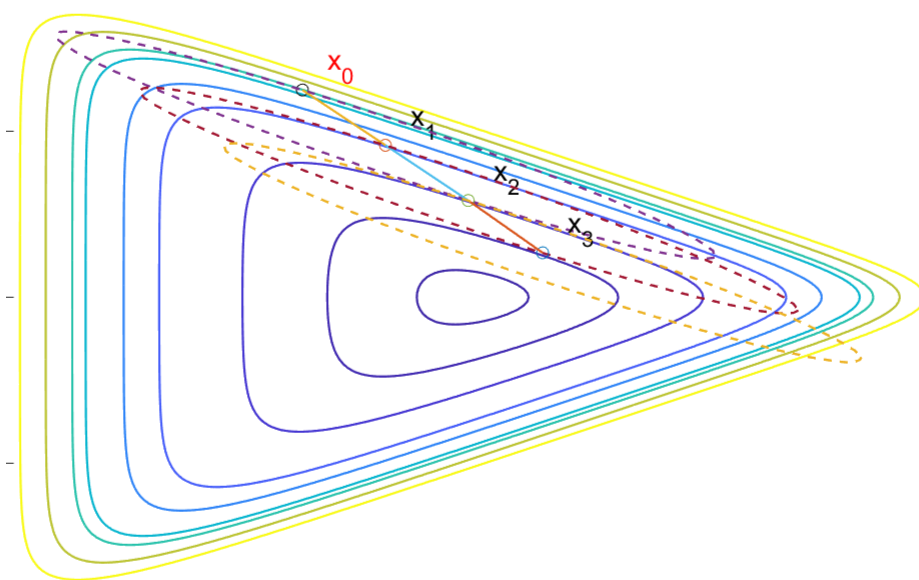


$$f(x) = e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} + e^{-x_1-0.1} \qquad x_0 = (-4,-2)$$

```
%x0=[-4;-2];
x0=[-2.4;2.5];
```



Example_NewtonMethod_Contours.m



牛顿方向与牛顿法

$$d^k = -H_k^{-1} g_k$$

① 使$f(x^{k+1})$在x^{k+1}处的取极小值的搜索方向$d^k = -H_k^{-1} g_k$称为f在x^k处的牛顿方向

② 每次迭代时都采用牛顿方向作为搜索方向的方法就称为牛顿法

牛顿法成立的前提条件 ① d^k充分小时

② f在x^k处的Hesse阵H_k非奇异

当f为凸二次函数时，同时满足①②，且二阶Taylor展开式精确成立

令步长为1，沿牛顿方向搜索一次就能找到最优点

牛顿方向的特点

✓ f在x_k处的牛顿方向是下降方向的充要条件是$H_k \succ 0$

证：如果取d_k为牛顿方向，$d_k = -H_k^{-1}g_k$，则

$$g_k^T d_k = -g_k^T H_k^{-1} g_k$$

$$d_k\text{是下降方向} \Leftrightarrow g_k^T d_k < 0 \Leftrightarrow g_k^T H_k^{-1} g_k > 0$$

$$\Leftrightarrow H_k^{-1} \succ 0 \quad \Leftrightarrow H_k \succ 0$$

✓ 牛顿法产生的迭代点序列在一定条件下是二阶收敛的

牛顿法在一定条件下是二阶收敛的

设函数$f(x)$二阶可导，x^*是局部极小点，且$H^{-1}(x^*)$存在. 如果牛顿法产生的迭代点列$\{x_k\}$充分接近x^*，且存在满足$k_1 k_2 < 1$的正常数k_1和k_2，对每一个$x_{k+1} \in \{x \mid \|x - x^*\| \le \|x_k - x^*\|\}$都有

$$\left\|H_{k+1}^{-1}\right\| \le k_1 \qquad \text{Hesse阵有界}$$

$$\frac{\|g^* - g_{k+1} - H_{k+1}(x^* - x_{k+1})\|}{\|x^* - x_{k+1}\|} \le k_2 \qquad \nabla f(x)\ \ L\text{连续}$$

则牛顿法产生的迭代点序列收敛于x^*. 并且，当牛顿法收敛时有下列关系：

$$\|x_{k+1} - x^*\| \le c\|x_k - x^*\|^2 \qquad c\text{是常数}$$

牛顿法的实现难点与不足

难点：

当 f 在 x_k 处的Hesse阵 $H_k \nsucc 0$ 时，不能保证牛顿方向 $-H_k^{-1}g_k$ 是下降方向

产生 →

- 阻尼牛顿法
- Levenberg-Marquardt牛顿法

不足：

✓ 计算Hesse矩阵的计算量较大

✓ 计算逆矩阵，或解线性方程组

$d_k = -H_k^{-1}g_k \qquad H_k d_k = -g_k$ → 产生拟牛顿法等

1.引言

2.梯度下降法

3.牛顿法

- 牛顿法
- **阻尼牛顿法**
- Levenberg-Marquardt牛顿法
- 高斯牛顿法

阻尼牛顿法

当 f 在 x_k 处的Hesse阵 $H_k \succ 0$ 时，牛顿方向 $-H_k^{-1}g_k$ 是下降方向

$$x_{k+1} = x_k - \alpha_k H_k^{-1} g_k$$

$$\alpha_k = \min_{\alpha \geq 0} f\left(x_k - \alpha H_k^{-1} g_k\right)$$

每次迭代都在方向 $-H_k^{-1}g_k$ 上开展一次一维搜索，由此确定每次搜索的步长.

由牛顿方向的性质知，阻尼牛顿法具备下降特性，即当 $g_k \neq 0$ 时，有

$$f_{k+1} < f_k$$

1.引言

2.梯度下降法

3.牛顿法

- 牛顿法
- 阻尼牛顿法
- Levenberg-Marquardt牛顿法
- 高斯牛顿法

修正的牛顿法

当$H_k \nsucc 0$时，将H_k修正为$\bar{H}_k \succ 0$，从而使修正的牛顿方向是下降方向

$f \in C^2 \rightarrow H_k$是实对称矩阵$\xrightarrow{\text{对角化}} H_k = U^T \Lambda U$

$\Lambda = \operatorname{diag}\big(\lambda_i(H_k)\big)$

U是对应的特征向量构成的正交矩阵，$U^T U = I$

设$H_k \nsucc 0$的最小特征值为$\lambda_{\min} < 0$　　构造 $\boxed{\bar{H}_k = H_k + (\varepsilon - \lambda_{\min}) I}$

ε是小正数，一般地，$\varepsilon = 0.01$

$$\begin{aligned}\bar{H}_k &= U^T \Lambda U + (\varepsilon - \lambda_{min}) U^T U \\ &= U^T \operatorname{diag}\big(\lambda_i(H_k) + (\varepsilon - \lambda_{min})\big)\, U\end{aligned}$$

$$\lambda_{\min}(\bar{H}_k) = \lambda_{\min} + (\varepsilon - \lambda_{min}) = \varepsilon > 0 \quad \Longrightarrow \bar{H}_k \succ 0$$

修正的牛顿方向 $d^k = -\bar{H}_k^{-1} g_k$是下降方向

A Modified Hessian

Algorithm 6.3 (Cholesky with Added Multiple of the Identity)

Choose $\beta = 10^{-3} > 0$;　$A \in S^n$

if $\min_i a_{ii} > 0$

　　set $\tau_0 \leftarrow 0$;

else

　　$\tau_0 = -\min h_{ii} + \beta$;

end (if)

for $k = 0,1,2,\ldots$

　　Attempt to apply the Cholesky algorithm to obtain $L^T L = A + \tau_k I$;

　　if the factorization is completed successfully

　　　　stop and return L;

　　else

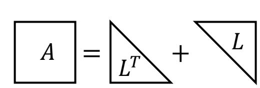
　　　　$\tau_{k+10} \leftarrow \max(2\tau_k, \beta)$;

　　end (if)

end (for)

$A \nsucc 0 \Longrightarrow A + \tau I \succ 0$

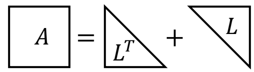
avoid eigenvalues



——'Numerical optimization'

Solve linear equations by Cholesky decomposition

$Ax = b \qquad A = L^T L$



$A \in S^n$

$$L^T(Lx) = b$$
$$Lx = y$$

$$L^T y = b \rightarrow \boxed{\begin{aligned} &y_1 = b_1/l_{11} \\ &y_i = \left(b_i - \sum_{k=1}^{i-1} l_{ki} y_k\right) \Big/ l_{ii},\ i = 2, \dots, n \end{aligned}}$$

$$Lx = y \rightarrow \boxed{\begin{aligned} &x_n = y_n/l_{nn} \\ &x_i = \left(y_i - \sum_{k=i+1}^{n} l_{ik} x_k\right) \Big/ l_{ii},\ i = n-1, n-2, \dots, 1 \end{aligned}}$$

Algorithm SolveLinEqn_Chol

牛顿法的算法

步骤1：已知目标函数$f(x)$，初始点x^0，$\varepsilon = 0.01$，精度要求$tol.\ k = 0$

步骤2：①计算$g_k = g(x^k)$，若$\|g_k\| < tol$，则终止迭代，输出x^k及$f(x^k)$

②计算$H_k = H(x^k)$；如果$H_k \nsucc 0$，$H_k := H_k + \varepsilon I \succ 0$ Algorithm 6.3 $H = L^T L$

③解方程组$H_k d_k = -g_k$

Algorithm SolveLinEqn_Chol $L^T L d = -g$

步骤3：从x^k出发，沿方向d^k进行一维非精确搜索得到可接受步长α_k

步骤4：计算新点$x^{k+1} = x^k + \alpha_k d^k$及$f(x^{k+1})$

步骤5：若$\|x^{k+1} - x^k\| < tol[\min\{1, \|\boldsymbol{x}^k\|\}]$，则终止迭代，输出$x^{k+1}$及$f(x^{k+1})$

否则，$k = k + 1$，转步骤2

牛顿法的MATLAB程序

Newton.m

Wolfe__Search.m

增加一个判定：梯度为零就是最优点，停止迭代

```
norm(g_current)<1.e-8
```

例

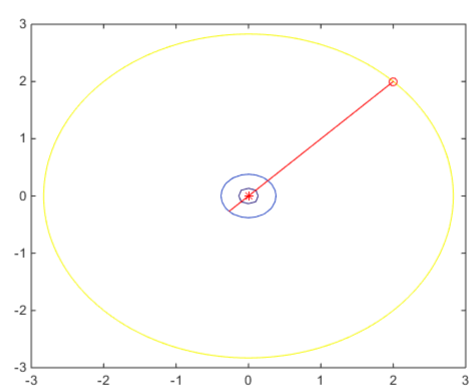
例5.4 用牛顿法求解多维无约束优化问题

（取初始点$x^0=(2,2), tol=1\times10^{-6}$）

$$\min f(x)=-\frac{1}{x_1^2+x_2^2+2}$$

example_5_4_CH05.m

x_optimal =1.0e-18 *[-0.1058   -0.1058]
f_optimal = -0.5000
k =5



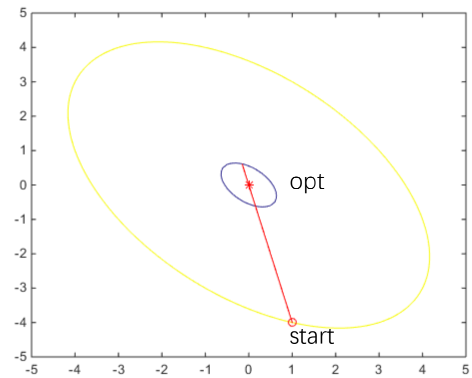
例5.5 用最速下降法求解多维无约束优化问题

（取初始点$x^0 = (1, -4), tol = 1 \times 10^{-6}$）

$$\min f(x) = x_1^2 + x_2^2 + x_1 x_2 + 2$$

example_5_5_CH05.m

x_optimal =1.0e-15 *[0.0833 -0.3331]
f_optimal = 2
k =3



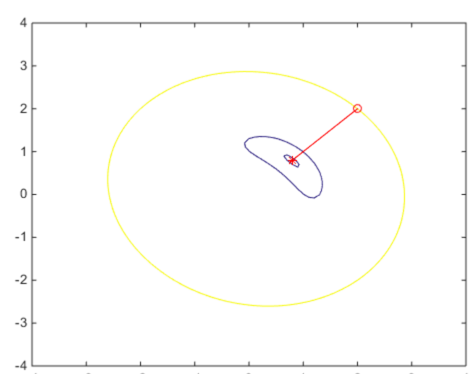
例5.6 用牛顿法求解多维无约束优化问题

（取初始点$x^0 = (2,2), tol = 1 \times 10^{-6}$）

$$\min f(x) = (x_1^2 + x_2^2 - 1)^2 + (x_1 + x_2 - 2)^2$$

example_5_6_CH05.m

x_optimal =[0.7937 0.7937]
f_optimal =0.2378
k =5



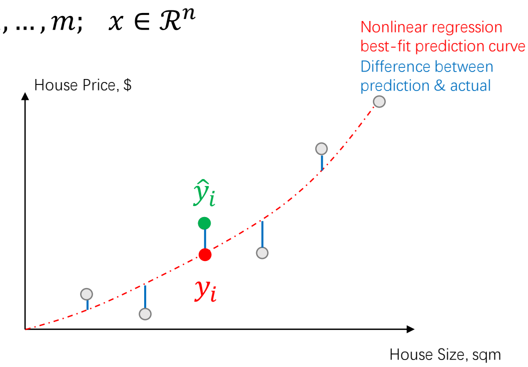
高斯牛顿法　引入

Residual Value = Observed or Measured Data − Predicted Data

r_i　　　$\hat{y}_i$　　　y_i

$$r(x) = \begin{bmatrix} r_1(x) \\ \vdots \\ r_m(x) \end{bmatrix} = \begin{bmatrix} \hat{y}_1 - y_1(x) \\ \vdots \\ \hat{y}_m - y_m(x) \end{bmatrix} \qquad y_i = y_i(x), i = 1, \ldots, m; \quad x \in \mathcal{R}^n$$

$$f(x) = \sum_{i=1}^{m} r_i^2(x) = [r_1 \ \cdots \ r_m] \begin{bmatrix} r_1 \\ \vdots \\ r_m \end{bmatrix} = r^T r = \|r\|^2$$



$\min f(x)$　　Predicted Data → Actual Data

高斯牛顿法的原理

$$\min f(x) = \sum_{i=1}^{m} \big(r_i(x)\big)^2$$

$x \in \mathcal{R}^n$

$r_i: \mathcal{R}^n \to \mathcal{R},\ i = 1, \cdots, m$, are given functions

$r_i(x) = \hat{y}_i - y_i(x)$

$$f(x) = r(x)^T r(x)$$

$$Df(x) = r(x)^T Dr(x) + r(x)^T Dr(x) = 2r(x)^T J(x) \quad (1 \times m) \times (m \times n) = 1 \times n$$

$$\nabla f(x) = 2J(x)^T r(x) \quad n \times 1$$

$$J(x) \triangleq Dr(x) = \begin{bmatrix} \frac{\partial r_1}{\partial x_1} & \cdots & \frac{\partial r_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial r_m}{\partial x_1} & \cdots & \frac{\partial r_m}{\partial x_n} \end{bmatrix}_{m \times n}$$

雅可比矩阵

求二阶导$D^2 f(x)$ $\quad d[\nabla f(x)] = D[\nabla f(x)]dx$

$$\nabla f(x) = 2J(x)^T r(x)$$

Using Total Differential Formula

$$J(x) \triangleq Dr(x) = \begin{bmatrix} \frac{\partial r_1}{\partial x_1} & \cdots & \frac{\partial r_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial r_m}{\partial x_1} & \cdots & \frac{\partial r_m}{\partial x_n} \end{bmatrix}_{m \times n}$$

$$d[r(x)] = Dr(x)dx = J(x)dx$$

$$\begin{aligned} d[\nabla f(x)] &= 2d[J(x)^T]r(x) + 2J(x)^T d[r(x)] \\ &= 2d[J(x)^T]r(x) + 2J(x)^T J(x)dx \\ &= 2[dJ(x)]^T r(x) + 2J(x)^T J(x)dx \in \mathcal{R}^{n\times 1} \end{aligned}$$

$$\begin{aligned} d[J] &= d\begin{bmatrix} \frac{\partial r_1}{\partial x_1} & \cdots & \frac{\partial r_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial r_m}{\partial x_1} & \cdots & \frac{\partial r_m}{\partial x_n} \end{bmatrix} \\ &= \frac{\partial J}{\partial x_1}dx_1 + \frac{\partial J}{\partial x_2}dx_2 + \cdots + \frac{\partial J}{\partial x_n}dx_n \end{aligned}$$

$$\frac{\partial J}{\partial x_j} = \begin{bmatrix} \frac{\partial^2 r_1}{\partial x_j \partial x_1} & \cdots & \frac{\partial^2 r_1}{\partial x_j \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 r_m}{\partial x_j \partial x_1} & \cdots & \frac{\partial^2 r_m}{\partial x_j \partial x_n} \end{bmatrix} \quad m \times n$$

求二阶导$D^2f(x)$

$$d[J] = \frac{\partial J}{\partial x_1}dx_1 + \frac{\partial J}{\partial x_2}dx_2 + \cdots + \frac{\partial J}{\partial x_n}dx_n$$

$$\begin{aligned}
[dJ]^T r &= \left[\frac{\partial J}{\partial x_1}dx_1 + \frac{\partial J}{\partial x_2}dx_2 + \cdots + \frac{\partial J}{\partial x_n}dx_n\right]^T r \\
&= \left[\frac{\partial J}{\partial x_1}^T dx_1 + \frac{\partial J}{\partial x_2}^T dx_2 + \cdots + \frac{\partial J}{\partial x_n}^T dx_n\right] r \\
&= \frac{\partial J}{\partial x_1}^T r dx_1 + \frac{\partial J}{\partial x_2}^T r dx_2 + \cdots + \frac{\partial J}{\partial x_n}^T r dx_n \\
&= \underbrace{\left[\frac{\partial J}{\partial x_1}^T r \quad \frac{\partial J}{\partial x_2}^T r \quad \cdots \quad \frac{\partial J}{\partial x_n}^T r\right]}_{n\times n} dx \\
&= S^T dx
\end{aligned}$$

$$d[J(x)^T]r(x) = S(x)^T dx$$

函数$r(x)$的二阶偏导数矩阵

$$\underset{n\times m}{\frac{\partial J}{\partial x_j}^T}\ \underset{m\times 1}{r} \qquad n\times 1$$

$$S(x) = \begin{bmatrix} r(x)^T \dfrac{\partial J(x)}{\partial x_1} \\ r(x)^T \dfrac{\partial J(x)}{\partial x_2} \\ \vdots \\ r(x)^T \dfrac{\partial J(x)}{\partial x_n} \end{bmatrix} \in \mathcal{R}^{n\times n}$$

$S(x)$含有二阶偏导数的矩阵

$$d[J(x)^T]r(x) = S(x)^T dx$$

$$\begin{aligned}
d[\nabla f(x)] &= 2d[J(x)^T]r(x) + 2J(x)^T J(x)dx \\
&= 2[S(x)^T + J(x)^T J(x)]dx \\
&= 2[S(x) + J(x)^T J(x)]^T dx
\end{aligned}$$

$$\nabla f(x) = 2J(x)^T r(x)$$

$$D^2 f(x) = D[\nabla f(x)] = 2[S(x) + J(x)^T J(x)]^T$$

$$\nabla^2 f(x) = 2[S(x) + J(x)^T J(x)]$$

$$H_f(x) = 2[S(x) + J(x)^T J(x)] \approx J(x)^T J(x)$$

$$\boxed{d[\nabla f(x)] = D[\nabla f(x)]dx}$$

二阶偏导数矩阵，$o(\delta^2)$，可略去

应用牛顿法，得

$$x^{k+1} = x^k - \Big(\overset{\succcurlyeq 0}{J(x^k)^T J(x^k)} + S(x^k)\Big)^{-1} J(x^k)^T r(x^k)$$

Newton’s algorithm reduces to the *Gauss-Newton method*:

$$x^{k+1} = x^k - \left(J(x^k)^T J(x^k)\right)^{-1} J(x^k)^T r(x^k)$$

高斯牛顿法的实现难点

当 f 在 x^k 处的Hesse矩阵 $H_f(x^k) \succcurlyeq 0$ 时，修正 $H_f(x^k)$ 为 $\bar{H}_f(x^k) \succ 0$ 以保证高斯牛顿法是下降迭代算法

设 $H_f(x^k)$ 的最小特征值为 $\lambda_{\min} = 0$，则修正 $H_f(x^k)$ 为 $\bar{H}_f(x^k)$

$$\bar{H}_f(x^k) = H_f(x^k) + \varepsilon I$$

ε 是一个较小的正数，一般地，$\varepsilon = 0.01$

$$H_f(x^k) \succcurlyeq 0 \longrightarrow \bar{H}_f(x^k) \succ 0$$

采用Algorithm 3.3，计算效率高

高斯牛顿法的计算步骤

步骤1：目标函数 $f(x) = [r(x)]^T r(x) = \sum_{p=1}^{m} [r_p(x)]^2$，初始点 x^0，精度 $tol.\ k = 0$

步骤2：①计算 $g_f(x^k)$，若 $\|g_f(x^k)\| < tol$，则终止迭代，输出 x^k 及 $f(x^k)$

②计算 $H_f(x^k)$；且，$H_f(x^k) := H_f(x^k) + \varepsilon I \succ 0$ Algorithm 6.3 $H = L^T L$

③解方程组 $H_f(x^k) d^k = -g_f(x^k)$ $\varepsilon = 1e-10$

Algorithm SolveLinEqn_Chol

步骤3：从 x^k 出发，沿方向 d^k 进行一维非精确搜索得到可接受步长 α_k

步骤4：计算新点 $x^{k+1} = x^k + \alpha_k d^k$ 及 $f(x^{k+1})$

步骤5：若 $\|x^{k+1} - x^k\| < tol[\min\{1, \|\boldsymbol{x}^k\|\}]$，则终止迭代，输出 x^{k+1} 及 $f(x^{k+1})$

否则，$k = k + 1$，转步骤2

高斯牛顿法的MATLAB程序

Guass_Newton.m

Wolfe__Search.m

Guass_Newton_Armijo.m

Armijo_mk.m

简单取

$$H_f(x^k) := \overline{H}_f(x^k) = H_f(x^k) + \varepsilon I$$

$$\varepsilon = 1^{-10}$$

迭代过程中，会出现Hesse矩阵 $det(H) = 0$ ，下降方向接近于0，无法搜索

采用Armijo搜索法效率高

实例测试

例5.7 用高斯牛顿法解方程组（初始点 $\boldsymbol{x}^0 = (2,2), tol = 1 \times 10^{-6}$ ）

$$\begin{cases} x_1^2 + x_2^2 = 1 \\ x_1 + x_2 = 2 \end{cases}$$

example_5_7_CH05.m

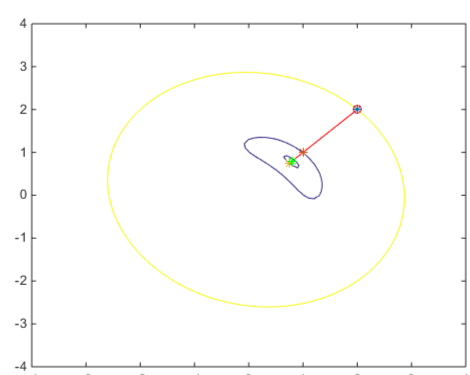
Guass_Newton.m

Wolfe__Search.m

解： $\min f(\boldsymbol{x}) = (x_1^2 + x_2^2 - 1)^2 + (x_1 + x_2 - 2)^2$

x_optimal =[0.7937   0.7937]
F_optimal = 0.2378
k = 5

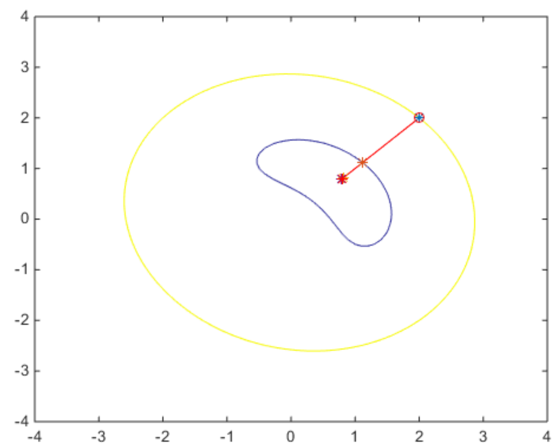
k	x^(k)		F^(k)
1	2.0000	2.0000	53.0000
2	0.9974	0.9974	0.9791
3	0.7487	0.7487	0.2673
4	0.7944	0.7944	0.2378
5	0.7937	0.7937	0.2378
6	0.7937	0.7937	0.2378



k	x^(k)		F^(k)
1	2.0000	2.0000	53.0000
2	1.1177	1.1177	2.3006
3	0.8188	0.8188	0.2475
4	0.7922	0.7922	0.2378
5	0.7938	0.7938	0.2378
6	0.7937	0.7937	0.2378
7	0.7937	0.7937	0.2378

testdata.txt

采用Armijo搜索法效率高



example_5_7_CH05_Armijo.m

Guass_Newton_Armijo.m

armijo_mk.m

例5.8 用高斯牛顿法解方程组（初始点$\boldsymbol{x}^0 = (4,13,1), tol = 1\times 10^{-6}$）

$$\begin{cases} x_1 = -5, x_2 = -8, x_3 = -7 \\ \sqrt{2}x_1x_2 = 0 \\ 2x_1x_3 = 0 \end{cases}$$

解： $\min f(\boldsymbol{x}) = (x_1+5)^2 + (x_2+8)^2 + (x_3+7)^2 + 2x_1^2x_2^2 + 4x_1^2x_3^2$

example_5_8_CH05.m

Guass_Newton.m

Wolfe__Search.m

x_optimal =[-0.0154; -7.9962; -6.9933]
F_optimal = 24.9230
k = 7

`example_5_8_CH05_Armijo.m`

`Guass_Newton_Armijo.m`

`armijo_mk.m`

x_optimal =[-0.0154; -7.9962; -6.9934]
F_optimal = 24.9230
k = 9

Example 9.2

Suppose, m measurements of a process are given at m points in time, in Figure9.2 ($m = 21$).
Let $t_1, \dots, t_m$ denote the measurement times and $\hat{y}_1, \dots, \hat{y}_m$ the measurement values.
Note that $t_1 = 0$ while $t_{21} = 10$. Fit a sinusoid to the measurement data.
The equation of the sinusoid is $y = A\sin(\omega t + \phi)$
with appropriate choices of the parameters A, ω, and ϕ.
To formulate the data-fitting problem, the objective function

$$\sum_{i=1}^{m} (\hat{y}_i - A\sin(\omega t_i + \phi))^2$$

representing the sum of the squared errors between the measurement values and the function values at the corresponding points in time.

Let $\boldsymbol{x} = [A, \omega, \phi]^T$ represent the vector of decision variables.
Therefore obtain a nonlinear least-squares problem with

$$r_i(\boldsymbol{x}) = \hat{y}_i - A\sin(\omega t_i + \phi)$$

Defining $\boldsymbol{r} = [r_1, \dots, r_m]^T$,
the objective function as $f(\boldsymbol{x}) = \boldsymbol{r}(\boldsymbol{x})^T \boldsymbol{r}(\boldsymbol{x})$.

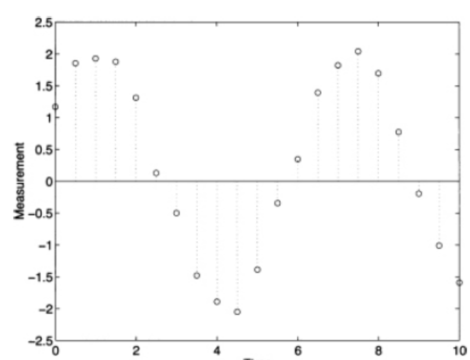


Figure 9.2 Measurement data for Example 9.2.

Example 9.2

Recall the data-fitting problem in Example 9.2, with

$$r_i(\boldsymbol{x}) = \hat{y}_i - A\sin(\omega t_i + \phi) \quad i = 1,2,\dots,21$$

The Jacobian matrix $J(\boldsymbol{x})$ in this problem is a 21×3 matrix with elements given by

$$\begin{aligned} \left(J(\boldsymbol{x})\right)_{(i,1)} &= -\sin(\omega t_i + \phi) \\ \left(J(\boldsymbol{x})\right)_{(i,2)} &= -t_i A\cos(\omega t_i + \phi) \qquad i = 1,2,\dots,21 \\ \left(J(\boldsymbol{x})\right)_{(i,3)} &= -A\cos(\omega t_i + \phi) \end{aligned}$$

Using the expressions above, we apply the Gauss-Newton algorithm to find the sinusoid of best fit, given the data pairs $(t_1, \hat{y}_1), \dots, (t_m, \hat{y}_m)$.

The parameters of this sinusoid are:

A = 2.01, ω = 0.992, and ϕ = 0.541.

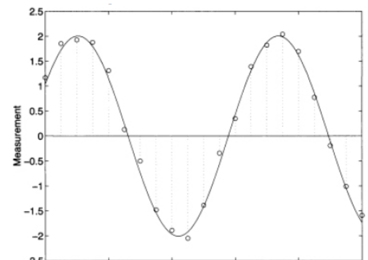
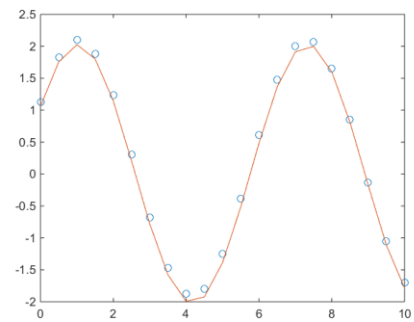


Figure 9.3 Sinusoid of best fit in Example 9.3



Example 9.2

P01_Chong_CH09_Example_9_3.m

Use 21 measurements $\hat{y}_i$ of a process are given at 21 points t_i in time

$\hat{y}_i = \mathrm{rand} + A\sin(\omega t_i + \phi) \quad i = 1,2,\dots,21$

$A = 2.01, \quad \omega = 0.992, \quad \phi = 0.541$

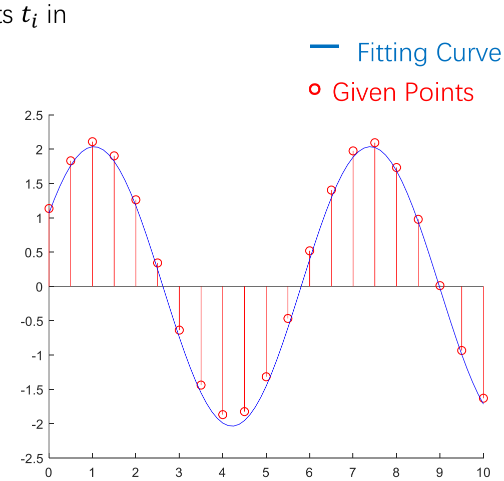
$t_i = \dfrac{10}{21-1}(i-1), \quad t_i \in [0,10]$

$r_i = 0.1$

rand = random number generator

Num. of iterations = 8

$\boldsymbol{x}^*$ =[2.0364 0.9874 0.5519]^T



P01_Chong_CH09_Example_9_3.m

```
close;
clear;
clc;
t1=0;t21=10;
N=21;
dt=(t21-t1)/(N-1);
A=2.;omig=1;phi=0.54;
ee=1.e-5;
for i=1:N
%suppose given points
    t(i)=(i-1)*dt;
%suppose random given erroes
    r0(i)= 0.1*rand;
%suppose given values
    y0(i)=r0(i)+A*sin(omig*t(i)+phi);
end

figure
hold on
stem(t,y0,'ro')
```

```
syms x1 x2 x3
%ri=yi-A*xin(omig*t+phi)
r=y0-x1*sin(x2*t+x3);
%J(x)=dr/dx
J=jacobian(r,[x1 x2 x3]);
%J(x)'
JT=transpose(J);
%J(x)'J(x)
F=JT*J;
%J(x)'*r(x)
JTr=JT*transpose(r);
%Change the text string to its function
funF=matlabFunction(F);
funJTr=matlabFunction(JTr);
```

rand()

Example 9.2

P01_Chong_CH09_Example_9_3.m

```
%max Num. Of iteration
M=200;
%Given an initial point
X(:,1)=[0.8;0.9;0.5];
%Given any different point for StopCond. ee=1.e-5;
X(:,2)=1.5*X(:,1);
dX=X(:,2)-X(:,1);
i=1;
while norm(dX)>ee & i<M
    x1=X(1,i);x2=X(2,i);x3=X(3,i);
%J'*J at (x1,x2,x3)
    F_v=funF(x1,x2,x3);
%(J'*J)^-1 at (x1,x2,x3)
    Inv_F_v=inv(F_v);
%J'*r at (x1,x2,x3)
    JTr_v=funJTr(x1,x2,x3);
%x(k+1)=x(k)-inv(J'*J)*J'*r
    X(:,i+1)=X(:,i)-Inv_F_v*JTr_v;
    dX=X(:,i+1)-X(:,i);
    i=i+1
End
%min A,omig and phi
fprintf('A =  %d , Omega = %d , Phi =  %d .\n',X(:,i));
```

$$\|\boldsymbol{x}^{(k+1)}-\boldsymbol{x}^{(k)}\|>\varepsilon$$

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}-\left(J(\boldsymbol{x})^TJ(\boldsymbol{x})\right)^{-1}J(\boldsymbol{x})^T\boldsymbol{r}(\boldsymbol{x})$$

```
%plot the fitting curve
dT=dt/4;
T=t1:dT:t21;
Y=X(1,i)*sin(X(2,i)*T+X(3,i));
plot(T,Y,'b')
```

作业

5-4

5-5

5-6

5-7