10. Unconstrained minimization

- terminology and assumptions
- gradient descent method
- steepest descent method
- Newton's method
- self-concordant functions
- implementation

Unconstrained minimization

simple box constraints allowed

minimize $f(x)$ s.t. $x \in [a,b]$

Mainly required for global optimality or convergence analysis (even local at times)

- f convex, twice continuously differentiable (hence $\mathbf{dom}\, f$ open)
- we assume optimal value $p^\star = \inf_x f(x)$ is attained (and finite)

unconstrained minimization methods

- produce sequence of points $x^{(k)} \in \mathbf{dom}\, f$, $k = 0, 1, \ldots$ with

$$f(x^{(k)}) \to p^\star \quad \text{or} \quad x^{(k)} \to x^\star$$

- can be interpreted as iterative methods for solving optimality condition

$$\nabla f(x^\star) = 0$$

or in case of non-diff fns should we insist that $0 \in \partial f(x^*)$ (subdifferential at x^*)?

Initial point and sublevel set

algorithms in this chapter require a starting point $x^{(0)}$ such that

- $x^{(0)} \in \mathbf{dom}\, f$: MUST
- sublevel set $S = \{x \mid f(x) \le f(x^{(0)})\}$ is closed

2nd condition is hard to verify, except when *all* sublevel sets are closed:

- equivalent to condition that $\mathbf{epi}\, f$ is closed
- true if $\mathbf{dom}\, f = \mathbf{R}^n$
- true if $f(x) \to \infty$ as $x \to \mathbf{bd}\,\mathbf{dom}\, f$

} sufficient conditions

examples of differentiable functions with closed sublevel sets:

$$f(x) = \log(\sum_{i=1}^{m} \exp(a_i^T x + b_i)), \qquad f(x) = -\sum_{i=1}^{m} \log(b_i - a_i^T x)$$

Strong convexity and implications

f is strongly convex on S if there exists an $m > 0$ such that

$$\nabla^2 f(x) \succeq mI \qquad \text{for all } x \in S$$

implications

- for $x, y \in S$,

$$f(y) \ge f(x) + \nabla f(x)^T (y - x) + \frac{m}{2}\|x - y\|_2^2 \ge f(x) - \frac{1}{2m}\|\nabla f(x)\|_2^2$$

  hence, S is bounded

- $p^\star > -\infty$, and for $x \in S$,

$$f(x) - p^\star \le \frac{1}{2m}\|\nabla f(x)\|_2^2$$

Recall this was proved earlier

useful as stopping criterion (if you know m)

We also proved: $\|x - x^*\| \le \frac{1}{m}\|\nabla f(x)\|_2$

Some running example of optimisation objectives for illustration: (which of them are convex?)

$x_1^* = 3 \quad x_2^* = 2 \quad f^* = 3$

(a) $f_a(x_1, x_2) = 3 + (x_1 - 1.5x_2)^2 + (x_2 - 2)^2$ s.t $x_1 \in [0,5]$, $x_2 \in [0,5]$

(b) $f_b(x_1, x_2) = 3 * [\sin(0.5 + 0.25 x_1 x_2)] \cos(x_1)$

(c) $f_c(x_1, x_2) = 100 [x_2 - x_1^2]^2 + (1 - x_1)^2$ } f1/w

(d) $f_d(x_1, x_2) = [1 + a - b x_1 - b x_2]_+^2 + [b + x_1 + a x_2 - b x_1 x_2]^2$

Try solving (minimizing each with cvx)

Eg: For (a) try http://www.cse.iitb.ac.in/~CS709/notes/code/cvx/function_a.m

```
cvx_begin
        variable x(2);
        minimize(3 + square(x(1) - 1.5*x(2)) + square(x(2)-2));
        subject to
        0 <= x(1);
        x(1) <= 5;
        0 <= x(2);
        x(2) <= 5;
cvx_end
```

```
>> function_a
Warning: A non-empty cvx problem already exists in this scope.
  It is being overwritten.
> In cvxprob.cvxprob at 27
  In cvx_begin at 41
  In function_a at 1

Calling SDPT3 4.0: 10 variables, 6 equality constraints

num. of constraints =  6
dim. of sdp    var  =  4,  num. of sdp  blk  =  2
dim. of linear var  =  4
*****************************************************************
  SDPT3: Infeasible path-following algorithms
```

Q: What kind of semi-definite program (a specific example of conic programs, i.e conic program with generalised inequality given by semi-definite cone) would you cast as? [complete what we began in class]

SDPT3's generic formulation of Semi-definite program

$$
\begin{aligned}
(P)\quad \min\ & \sum_{j=1}^{n_s}[\langle c_j^s, x_j^s\rangle - \nu_j^s \log\det(x_j^s)] + \sum_{i=1}^{n_q}[\langle c_i^q, x_i^q\rangle - \nu_i^q \log\gamma(x_i^q)] \\
& + \langle c^l, x^l\rangle - \sum_{k=1}^{n_l}\nu_k^l \log x_k^l + \langle c^u, x^u\rangle \\
\text{s.t.}\ & \sum_{j=1}^{n_s}\mathcal{A}_j^s(x_j^s) + \sum_{i=1}^{n_q}A_i^q x_i^q + A^l x^l + A^u x^u = b, \\
& x_j^s \in K_s^{s_j}\ \forall j,\quad x_i^q \in K_q^{q_i}\ \forall i,\quad x^l \in K_l^{n_l},\quad x^u \in \mathbb{R}^{n_u}.
\end{aligned}
$$

Here, c_j^s, x_j^s are symmetric matrices of dimension s_j and $K_s^{s_j}$ is the cone of positive semidefinite symmetric matrices of the same dimension. Similarly, c_i^q, x_i^q are vectors in $\mathbb{R}^{q_i}$ and $K_q^{q_i}$ is the quadratic or second-order cone defined by $K_q^{q_i} := \{x = [x_0; \bar{x}] \in \mathbb{R}^{q_i} : x_0 \geq \sqrt{\bar{x}^T\bar{x}}\}$. Finally, c^l, x^l are vectors of dimension n_l, $K_l^{n_l}$ is the nonnegative orthant $\mathbb{R}_+^{n_l}$, and c^u, x^u are vectors of dimension n_u. In the notation above, $\mathcal{A}_j^s$ is the linear map from $K_s^{s_j}$ to $\mathbb{R}^m$ defined by

$$\mathcal{A}_j^s(x_j^s) = \left[\langle a_{j,1}^s, x_j^s\rangle; \ldots; \langle a_{j,m}^s, x_j^s\rangle\right],$$

where $a_{j,1}^s, \ldots, a_{j,m}^s \in \mathcal{S}^{s_j}$ are constraint matrices associated with the jth semidefinite block variable x_j^s. The matrix A_i^q is an $m \times q_i$ dimensional constraint matrix corresponding to the ith quadratic block variable x_i^q, and A^l and A^u are the $m \times n_l$ and $m \times n_u$ dimensional constraint matrices corresponding to the linear block variable x^l and the unrestricted block variable x^u. The notation $\langle p, q\rangle$ denotes the standard inner product in the appropriate space. For a given vector $u = [u_0; \bar{u}]$ in a second order cone, we define $\gamma(u) := \sqrt{u_0^2 - \bar{u}^T\bar{u}}$. In the problem (P), ν_j^s, ν_i^q, and ν_k^l are given nonnegative parameters.

What we began in class:

Original: $\min_{x_1, x_2} \; 3 + (x_1 - 1.5x_2)^2 + (x_2 - 2)^2$ — 2 variables, 4 constraints

s.t. $0 \le x_1 \le 5 \qquad 0 \le x_2 \le 5$

Solns are equivalent

Goal: mapping into a standard SDP

SPD: $\min_{x_1, x_2, t, \alpha_i} \; t$

s.t. $t - \left(3 + (x_1 - 1.5x_2)^2 + (x_2 - 2)^2\right) = \alpha_1 \qquad \alpha_1 \ge 0$

$x_1 \ge 0 \qquad x_2 \ge 0 \qquad 5 - x_1 = \alpha_2 \qquad \alpha_2 \ge 0 \qquad 5 - x_2 = \alpha_3 \qquad \alpha_3 \ge 0$

6 variables, 8 constraints

Descent methods

$$x^{(k+1)} = x^{(k)} + t^{(k)}\Delta x^{(k)} \quad \text{with } f(x^{(k+1)}) < f(x^{(k)})$$

- other notations: $x^+ = x + t\Delta x$, $x := x + t\Delta x$
- Δx is the *step*, or *search direction*; t is the *step size*, or *step length*
- from convexity, $f(x^+) < f(x)$ implies $\nabla f(x)^T \Delta x < 0$ (*i.e.*, Δx is a *descent direction*)

General descent method.

given a starting point $x \in \mathbf{dom}\, f$.
repeat
1. Determine a descent direction Δx.
2. *Line search.* Choose a step size $t > 0$.
3. *Update.* $x := x + t\Delta x$.

until stopping criterion is satisfied.

Line search types

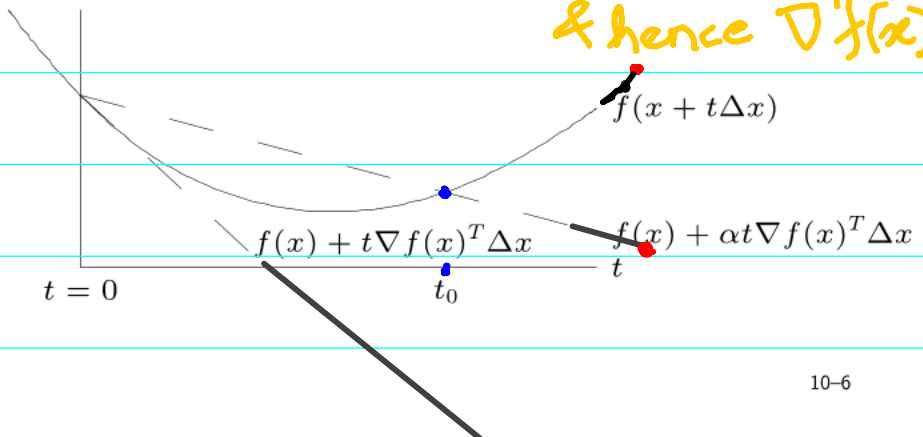
exact line search: $t = \operatorname{argmin}_{t>0} f(x + t\Delta x)$

backtracking line search (with parameters $\alpha \in (0, 1/2)$, $\beta \in (0, 1)$)

- starting at $t = 1$, repeat $t := \beta t$ until

$$f(x + t\Delta x) < f(x) + \alpha t \nabla f(x)^T \Delta x$$

- graphical interpretation: backtrack until $t \le t_0$



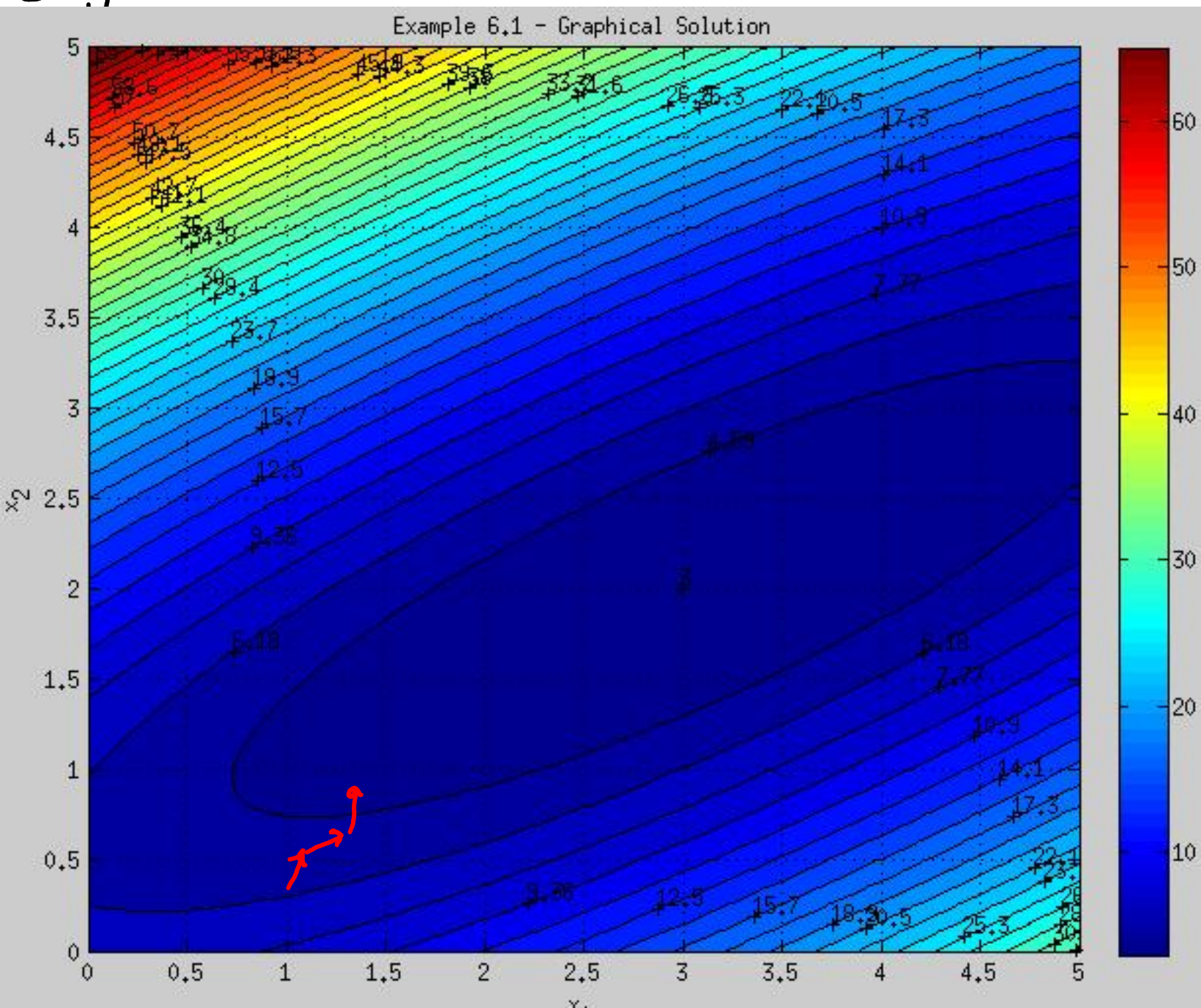
$f(x) + t\nabla f(x)\Delta x$

Can we understand effect of choice of Δx & t with respect to level curves?

http://www.cse.iitb.ac.in/~CS709/notes/code/unconstrainedOpt/GraphicalSolution_Example6_1a.m

Minimize $f(x_1, x_2) = 3 + (x_1 - 1.5x_2)^2 + (x_2 - 2)^2$

Subject to: $0 \le x_1 \le 5$ $0 \le x_2 \le 5$



In summary [from http://www.cse.iitb.ac.in/~cs709/notes/BasicsOfConvexOptimization.pdf pages 300 & 301]

Find a starting point $\mathbf{x}^{(0)} \in \mathcal{D}$
repeat
1. Determine $\Delta\mathbf{x}^{(k)}$. (Descent algos: $\nabla f(x^{(k)})^T \Delta x^k < 0$)
2. Choose a step size $t^{(k)} > 0$ using ray[a] search.
3. Obtain $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + t^{(k)}\Delta\mathbf{x}^{(k)}$.
4. Set $k = k + 1$.

until stopping criterion (such as $||\nabla f(\mathbf{x}^{(k+1)})|| < \epsilon$) is satisfied → other stopping criteria

[a]Many textbooks refer to this as line search, but we prefer to call it ray search, since the step must be positive.

Figure 4.45: The general descent algorithm.

1. **Exact ray search:** The exact ray search seeks a scaling factor t that satisfies

$$t = \underset{t>0}{\operatorname{argmin}} f(\mathbf{x} + t\Delta\mathbf{x}) \tag{4.89}$$

2. **Backtracking ray search:** The exact line search may not be feasible or could be expensive to compute for complex non-linear functions. A relatively simpler ray search iterates over values of step size starting from 1 and scaling it down by a factor of $\beta \in (0, \frac{1}{2})$ after every iteration till the following condition, called the *Armijo condition* is satisfied for some $0 < c_1 < 1$.

At $c_1 = 1$, ...

Guaranteed decrease in $f(x+t\Delta x)$ and avoid OVERSTEPPING

$$f(\mathbf{x} + t\Delta\mathbf{x}) < \boxed{f(\mathbf{x}) + c_1 t\nabla^T f(\mathbf{x})\Delta\mathbf{x}} \tag{4.90}$$

Guaranteed decrease in magnitude of slope and avoid TOO SMALL STEPS

$$\left|\Delta\mathbf{x}^T \nabla f(\mathbf{x} + t\Delta\mathbf{x})\right| \leq \boxed{c_2 \left|\Delta\mathbf{x}^T \nabla f(\mathbf{x})\right|} \tag{4.91}$$

STRONG WOLFE CONDITIONS

where $1 > c_1 > c_2 > 0$. This condition ensures that the slope of the function $f(\mathbf{x}+t\Delta\mathbf{x})$ at t is less than c_2 times that at $t = 0$. The conditions in (4.90) and (4.91) are together called the strong Wolfe conditions.

Exercise: Let $f:\mathbb{R}\to\mathbb{R}$ and $f(x)=x^2$

Let $x^{(0)}=2$ $\Delta x^{(k)}=-1$ $\forall k$ (it is always a descent direction)

and $x^{(k)}=1+2^{-k}$

$$\Rightarrow t^{(k)}=\frac{x^{(k+1)}-x^{k}}{\Delta x^{(k)}}=2^{-k}-2^{-k-1}=2^{-k-1}$$

→ too diminishing a step size (see the problem it creates!)

Now:

$$f(x^{(k+1)})=\underbrace{(1+2^{-k-1})^2}_{f(x^{(k+1)})}<\underbrace{(1+2^{-k})^2}_{f(x^{(k)})}-\underbrace{c_1\cdot 2^{-k-1}\cdot 2^{-k+1}}_{c_1 t^{(k)}\nabla f(x^{(k)})\Delta x^{k}}\quad(*)$$

[Find c_1 for which above inequality holds]
[In practice, $c_1\in(0,1/2)$ and often $c_1=10^{-4}$]

Ans: One possible c_1 is $3/4$ (in fact any $c_1\le\frac{3}{4}$)

General claim:

It can be proved that $\exists c_1, c_2$ & $t^{(k)}$ s.t (4.90) and (4.91) are both satisfied for any f.

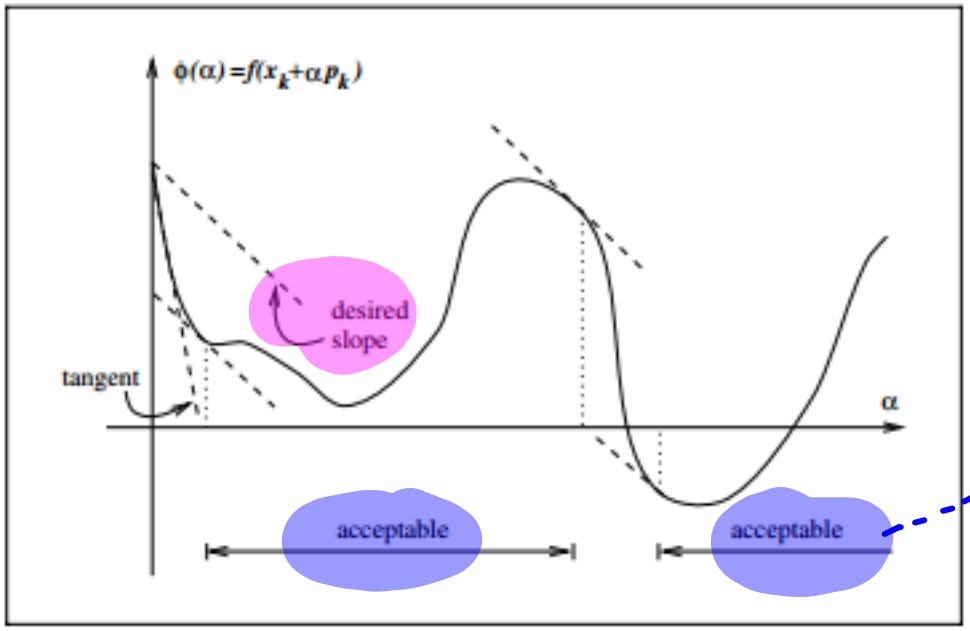
However, the problem is that though $(*)$ is satisfied,

(i) $\lim_{k\to\infty} x^{(k)}=1$

(ii) $\hat{x}_{opt}=$ 0

(iii) Even though f is convex & Wolfe cond (4.90) holds!

∴ (4.91) is important



Desired slope is sufficient curvature condition (avoid under-stepping) Illustrating (4.91)

CLAIM: If $f: D \to \mathbb{R}$ is continuously differentiable on D and f is bounded below along $\{x^{(k)} + t\Delta x^{(k)} \mid t > 0\}$. Then if $0 < c_1 < c_2 < 1$, there exist intervals of step lengths $t^{(k)}$ satisfying the Wolfe conditions (4.90) and (4.91)

H/W: Prove this. HINT: Make use of the mean value theorem

Other conditions:

- Goldstein: $f(x^{(k)}) + (1-c)t^{(k)}\nabla f(x^{(k)})^T \Delta x^{(k)} \leq f(x^{(k)} + t^{(k)}\Delta x^{(k)})$ (large step size)

$\leq f(x^{(k)}) + c t^{(k)} \nabla f(x_k)^T \Delta x^{(k)}$ → significant decrease

Proof of claim on previous page:

As stated, the function

$$\phi(t) = f(x^k + t\Delta x^k)$$ is bounded below for all $t > 0$.

Since $0 < c_1 < 1$, the linear approximation

$$l(t) = f(x^k) + t c_1 \nabla^T f(x^k) \Delta x^k$$

is unbounded below and must therefore intersect the graph of ϕ atleast once.

Let $t' > 0$ be the smallest intersecting value of t, that is:

$$f(x^k + t'\Delta x^k) = f(x^k) + t' c_1 \nabla^T f(x^k) \Delta x^k$$

Then for all $t \in [0, t']$,

$$f(x^k + t\Delta x^k) \leq f(x^k) + t c_1 \nabla^T f(x^k) \Delta x^k. \quad \text{(i)}$$

$(\because \nabla^T f(x^k) \Delta x^k < 0)$

This shows that there exists a non-empty set of t s.t First wolfe condition is met

By the mean value theorem, $\exists\ t'' \in (0, t')$ s.t

$$f(x^k + t'\Delta x^k) - f(x^k) = t' \nabla f(x^k + t''\Delta x^k)^T \Delta x^k. \quad \text{(ii)}$$

Combining (i) and (ii)

$$\nabla^T f(x^k + t'' \Delta x^k)\, \Delta x^k = c_1 \nabla^T f(x^k) \Delta x^k$$
$$> c_2 \nabla^T f(x^k) \Delta x^k$$

$(\because c_1 < c_2 \;\&\; \nabla^T f(x^k) \Delta x^k < 0)$

same reason

$$\Rightarrow \left| \nabla^T f(x^k + t'' \Delta x^k) \Delta x^k \right| < c_2 \left| \nabla^T f(x^k) \Delta x^k \right| \rightarrow \text{(iii)}$$

$\Rightarrow \exists\ t'' = t^k$ which satisfies Wolfe conditions (i) & (iii)

(and optionally, you can show that by smoothness of f, $\exists$ an interval around t'' for which (i) & (iii) hold)

Convergence analysis for general descent methods [without refering to convexity]

f is L-Lipschitz iff $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x-y\|_2$

http://www.math.jyu.fi/research/reports/rep100.pdf

$\forall\, x, y \in \mathbb{R}^n$

Let F_L be the class of L-Lipschitz functions

A (strong) form of (uniform) continuity of functions

Claim: Consider the descent algo in figure 4.45 with $x^{(k+1)} = x^{(k)} + t^{(k)}\Delta x^{(k)}$ where $\Delta x^{(k)}$ is a descent direction and $t^{(k)}$ satisfies the Wolfe conditions. Suppose f is bounded below in $\mathbb{R}^n$ and that f is continuously differentiable in an open set N containing the level set $\{x : f(x) \leq f(x^{(0)})\}$ where $x^{(0)}$ is the starting point of the iteration. Also, let $f \in F_L$. Then

$$\sum_{k=0}^{\infty} \frac{\left(\nabla^T f(x^{(k)})\Delta x^{(k)}\right)^2}{\|\Delta x^{(k)}\|^2} \text{ is finite ie } < \infty$$

Proof:

Since $\Delta x^{(k)}$ is a descent direction,

$$\nabla^T f(x^{(k)}) \Delta x^{(k)} < 0$$

and since

$$\left| \nabla^T f(x^{(k)} + t^{(k)} \Delta x^{(k)}) \Delta x^{(k)} \right| \leq c_2 \left| \underbrace{\nabla^T f(x^{(k)}) \Delta x^{(k)}}_{\text{since } < 0} \right|$$

it must be that (since $c_2 > 0$)

$$\nabla^T f(x^{(k)} + t^{(k)} \Delta x^{(k)}) \Delta x^{(k)} \geq c_2 \nabla^T f(x^{(k)}) \Delta x^{(k)}$$

Subtracting $\nabla^T f(x^{(k)}) \Delta x^{(k)}$ from both sides

$$\left[\nabla^T f(x^{(k)} + t^{(k)} \Delta x^{(k)}) - \nabla^T f(x^{(k)}) \right] \Delta x^{(k)} \geq (c_2 - 1) \nabla^T f(x^{(k)}) \Delta x^{(k)} \quad \text{(i.)}$$

Also

$$\left[\nabla^T f(x^{(k)} + t^{(k)} \Delta x^{(k)}) - \nabla^T f(x^{(k)}) \right] \Delta x^{(k)}$$

$$\leq \left\| \nabla f(x^{(k)} + t^{(k)} \Delta x^{(k)}) - \nabla f(x^{(k)}) \right\| \left\| \Delta x^{(k)} \right\| \quad \{\text{Cauchy Schwarz ineq}\}$$

$$\leq L \left\| \Delta x^{(k)} \right\|^2 t^{(k)} \quad \{\text{From L-Lipschitz condition}\} \quad \text{(ii)}$$

Combining (i) and (ii), < 0

$$t^{(k)} \geq \frac{c_2 - 1}{L} \cdot \frac{\nabla^T f(x^{(k)}) \Delta x^{(k)}}{\|\Delta x^{(k)}\|^2} \quad \text{(iii)}$$

($c_2 - 1 < 0$)

Substituting (iii) into the first Wolfe condition,

$$f(x^{(k+1)}) \leq f(x^{(k)}) - c_1 \frac{(1-c_2)}{L} \frac{\left(\nabla^T f(x^{(k)}) \Delta x^{(k)}\right)^2}{\|\Delta x^{(k)}\|^2}$$

Let this be c

(Applying this inq. recursively)

$$f(x^{(k+1)}) \leq f(x^{(0)}) - c \sum_{i=0}^{k} \frac{\left(\nabla^T f(x^{(i)}) \Delta x^{(i)}\right)^2}{\|\Delta x^{(i)}\|^2} \quad \text{(iv)}$$

[We assumed that f is bounded below — a very reasonable assumption, given we want to minimize f]

∴ Taking limits of (iv) as $k \to \infty$

$$\lim_{k\to\infty} c \sum_{i=0}^{k} \frac{\left(\nabla^T f(x^{(i)}) \Delta x^{(i)}\right)^2}{\|\Delta x^{(i)}\|^2} \leq \lim_{k\to\infty} f(x^{(k+1)}) - f(x^{(0)})$$

$< \infty$

Thus:

$$\sum_{k=0}^{\infty} \frac{\left(\nabla^T f(x^{(k)}) \Delta x^{(k)}\right)^2}{\|\Delta x^{(k)}\|^2} \quad \text{is} \quad \text{finite i.e.} < \infty \quad \rightarrow (v)$$

Hence proved

(v) implies that $\frac{\nabla^T f(x^{(k)}) \Delta x^{(k)}}{\|\Delta x^{(k)}\|} \rightarrow 0$ (vi)

If we additionally ensure that the descent direction is NOT ORTHOGONAL TO THE GRADIENT

i.e., $$\frac{-\nabla^T f(x^{(k)}) \Delta x^{(k)}}{\|\Delta x^{(k)}\| \, \|\nabla f(x^{(k)})\|} \geq r > 0 \quad \forall \, k \qquad \text{(vii)}$$

"minus" gives you abs value of $\nabla^T f(x^k) \Delta x^k$ which is < 0

then from (vi) and (vii)

$$\lim_{k \rightarrow \infty} \|\nabla f(x^{(k)})\| = 0$$

We will explain this for individual choices of descent directions

Similar analysis & proof holds for Goldstein conditions → H/W exercise

Steepest descent method

normalized steepest descent direction (at x, for norm $\|\cdot\|$):

$$\Delta x_{\text{nsd}} = \text{argmin}\{\nabla f(x)^T v \mid \|v\| = 1\}$$

prevent $\nabla f(x)^T v \to -\infty$

interpretation: for small v, $f(x+v) \approx f(x) + \nabla f(x)^T v$;
direction Δx_{nsd} is unit-norm step with most negative directional derivative

(unnormalized) steepest descent direction

$$\Delta x_{\text{sd}} = \|\nabla f(x)\|_* \Delta x_{\text{nsd}}$$

Recall: $\|y\|_x = \max_x x^T y$ s.t. $\|x\| \le 1$

dual norm

primal norm

H/W: verify

satisfies $\nabla f(x)^T \Delta_{\text{sd}} = -\|\nabla f(x)\|_*^2$

steepest descent method

- general descent method with $\Delta x = \Delta x_{\text{sd}}$
- convergence properties similar to gradient descent

Q: is condition (iii) satisfied?

Yes: $-\nabla^T f(x) \Delta x = 1 > 0$

$\|\nabla f(x)\| \|\Delta x\|$

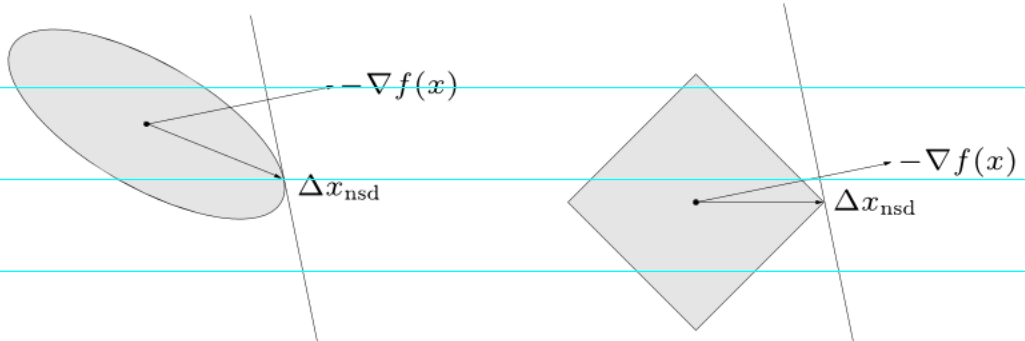
examples

A → • Euclidean norm: $\Delta x_{\text{sd}} = -\nabla f(x)$ ($\|\cdot\|_2$ is its own dual) (Gradient descent)

B → • quadratic norm $\|x\|_P = (x^T P x)^{1/2}$ ($P \in \mathbf{S}^n_{++}$): $\Delta x_{\text{sd}} = -P^{-1}\nabla f(x)$

C → • ℓ_1-norm: $\Delta x_{\text{sd}} = -(\partial f(x)/\partial x_i) e_i$, where $|\partial f(x)/\partial x_i| = \|\nabla f(x)\|_\infty$ ($\|\cdot\|_\infty$ is dual of $\|\ \|_1$) (coordinate descent)

unit balls and normalized steepest descent directions for a quadratic norm and the ℓ_1-norm:



D → Q: What is steepest descent for l_∞ norm?

Ans: $\Delta x_{sd} =$

Claim: for C which is called **COORDINATE DESCENT**

$$\Delta x_{nsd} = \underset{v}{\operatorname{argmin}} \{ \nabla^T f(x) v \mid \|v\|_1 = 1 \} = -\operatorname{sgn}\left(\partial f(x)/\partial x_i\right) e_i$$

where $|\partial f(x)/\partial x_i| = \|\nabla f(x)\|_\infty = \max_j |\partial f(x)/\partial x_j|$

Proof: (Denote $\partial f(x)/\partial x_i$ by $\nabla f(x)_i$)

① $\sum_{j=1}^n |(\nabla f(x))_j v_j| \leq \|\nabla f(x)\|_\infty \|v\|_1$ (See above for what $\|\nabla f(x)\|_\infty$ is)

$\leq \|\nabla f(x)\|_\infty$ (since $\|v\|_1 = 1$)

② Since $\nabla^T f(x) v \geq -\sum_{j=1}^n |(\nabla f(x))_j v_j|$, from ①

$$\nabla^T f(x) v \geq -\|\nabla f(x)\|_\infty = -|(\nabla f(x))_i|$$

same index i

$$= -\max_j |(\nabla f(x))_j|$$

③ If $v = -\operatorname{sgn}\left((\nabla f(x))_i\right) e_i$ then $\nabla^T f(x) v = -|(\nabla f(x))_i| = -\|\nabla f(x)\|_\infty$

That is, $\exists$ choice of v for which lower bound on $\nabla^T f(x) v$ in ② is actually attained!

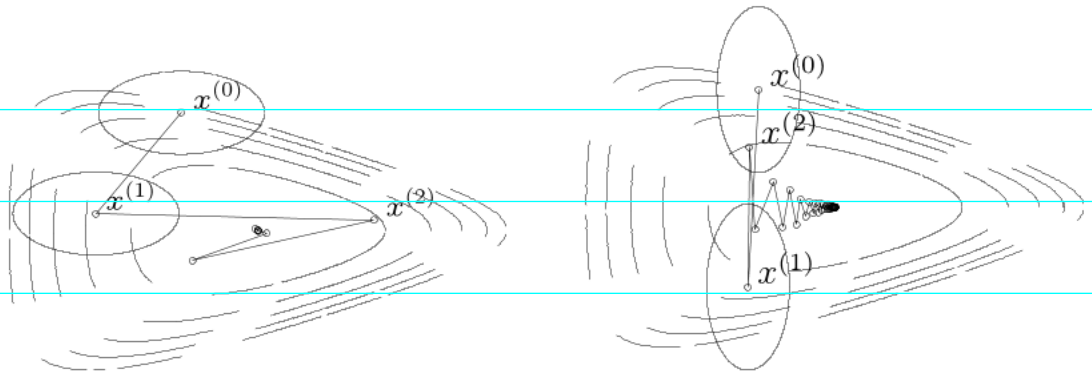
④ Therefore

$$\Delta x_{nsd} = -\operatorname{sgn}\left(\nabla f(x)_i\right) e_i$$

$$\Delta x_{sd} = -\left(\nabla f(x)_i\right) e_i$$

verify that $\Delta x_{sd} = \|\nabla f(x)\|_* \Delta x_{nsd}$ & $\nabla^T f(x) \Delta x_{sd} = -\|\nabla f(x)\|_*^2$

choice of norm for steepest descent



- steepest descent with backtracking line search for two quadratic norms
- ellipses show $\{x \mid \|x - x^{(k)}\|_P = 1\}$
- equivalent interpretation of steepest descent with quadratic norm $\|\cdot\|_P$: gradient descent after change of variables $\bar{x} = P^{1/2}x$

shows choice of P has strong effect on speed of convergence

Gradient descent method

general descent method with $\Delta x = -\nabla f(x)$

---

given a starting point $x \in \mathbf{dom}\, f$.
repeat
1. $\Delta x := -\nabla f(x)$.
2. *Line search.* Choose step size t via exact or backtracking line search.
3. *Update.* $x := x + t\Delta x$.

until stopping criterion is satisfied.

- stopping criterion usually of the form $\|\nabla f(x)\|_2 \le \epsilon$ → typical test
- convergence result: for strongly convex f,

$$f(x^{(k)}) - p^\star \le c^k (f(x^{(0)}) - p^\star)$$

$c \in (0,1)$ depends on m, $x^{(0)}$, line search type

- very simple, but often very slow; rarely used in practice

$\frac{f(x^k) - p^*}{f(x^{k-1}) - p^*} \to c$

H/w: Modify code at http://www.cse.iitb.ac.in/~CS709/notes/code/unconstrainedOpt/SteepestDescent_Example6_1a.m to try gradient descent on the Rosenbrock function:

$f(x_1, x_2) = (1-x_1)^2 + 100(x_2 - x_1^2)^2$

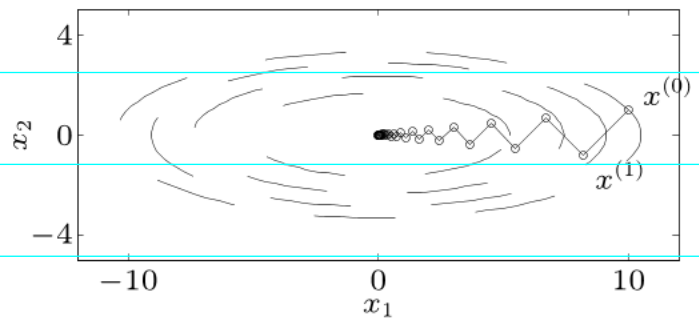
quadratic problem in $\mathbf{R}^2$

$$f(x) = (1/2)(x_1^2 + \gamma x_2^2) \qquad (\gamma > 0)$$

with exact line search, starting at $x^{(0)} = (\gamma, 1)$:

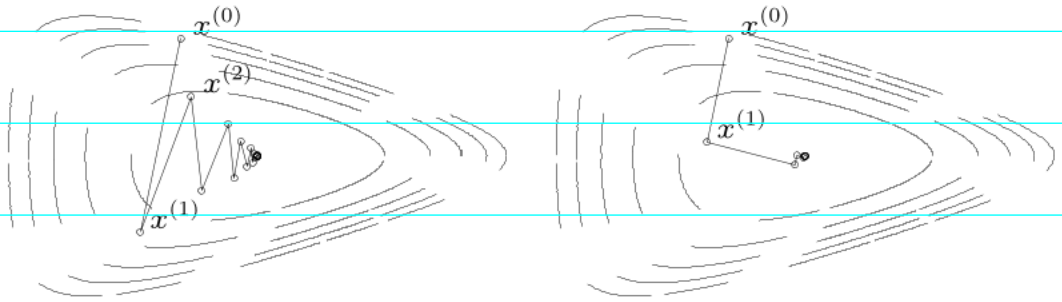
$$x_1^{(k)} = \gamma \left(\frac{\gamma - 1}{\gamma + 1}\right)^k, \qquad x_2^{(k)} = \left(-\frac{\gamma - 1}{\gamma + 1}\right)^k$$

- very slow if $\gamma \gg 1$ or $\gamma \ll 1$
- example for $\gamma = 10$:



nonquadratic example

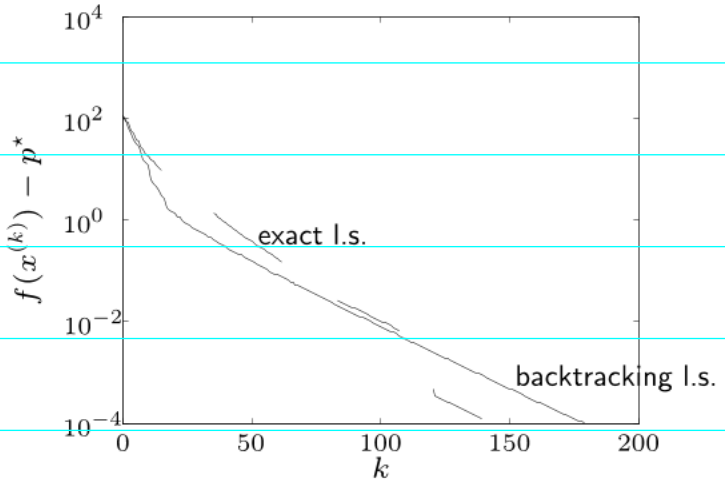
$$f(x_1, x_2) = e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} + e^{-x_1-0.1}$$



backtracking line search exact line search

a problem in $\mathbf{R}^{100}$

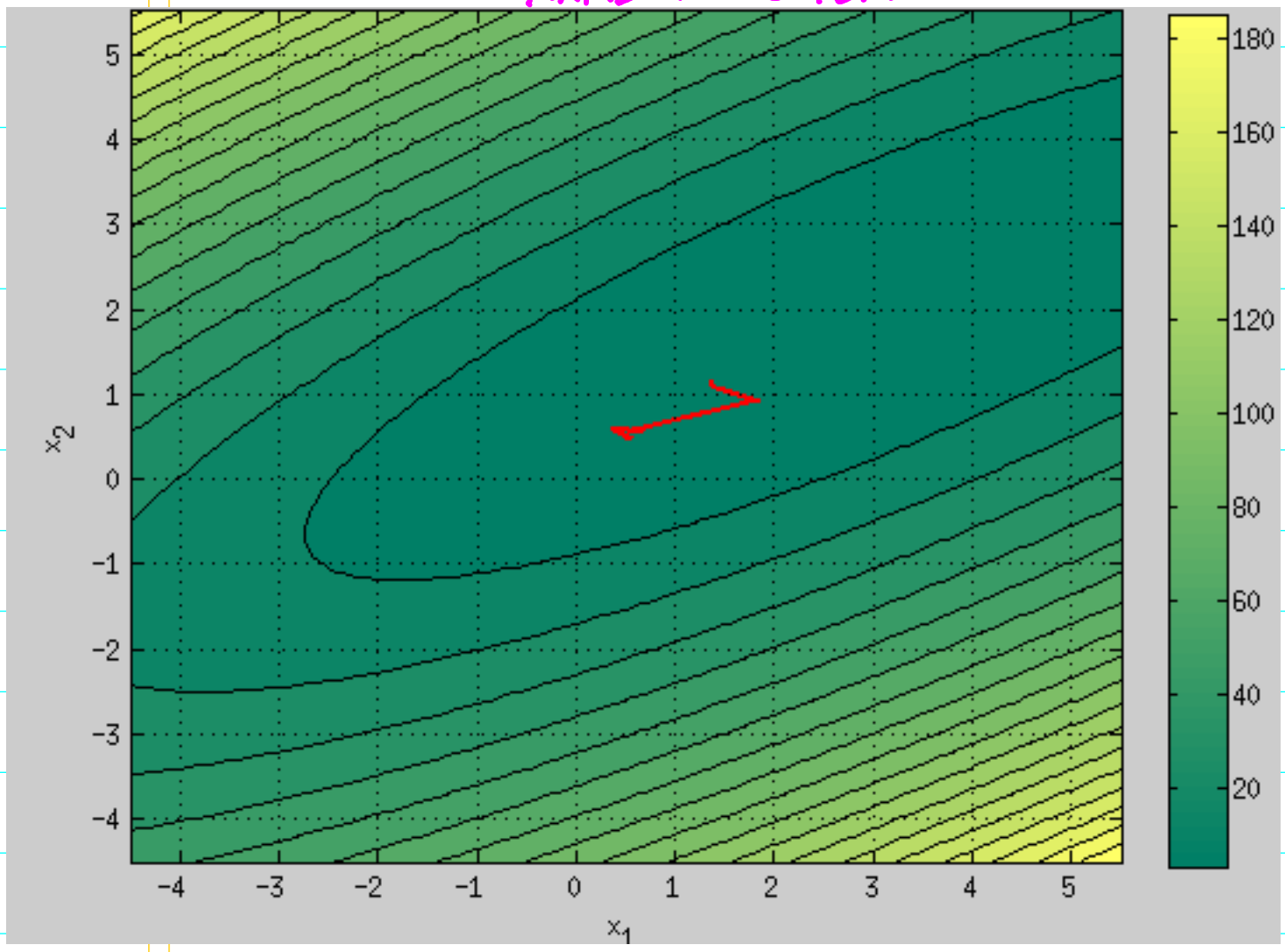
$$f(x) = c^T x - \sum_{i=1}^{500} \log(b_i - a_i^T x)$$



'linear' convergence, *i.e.*, a straight line on a semilog plot

Contrast running of gradient descent with random walk on objectives (a) (b) (c) & (d) mentioned earlier

For (a) ⇒ http://www.cse.iitb.ac.in/~CS709/notes/code/unconstrainedOpt/RandomWalk_Example6_1a.m

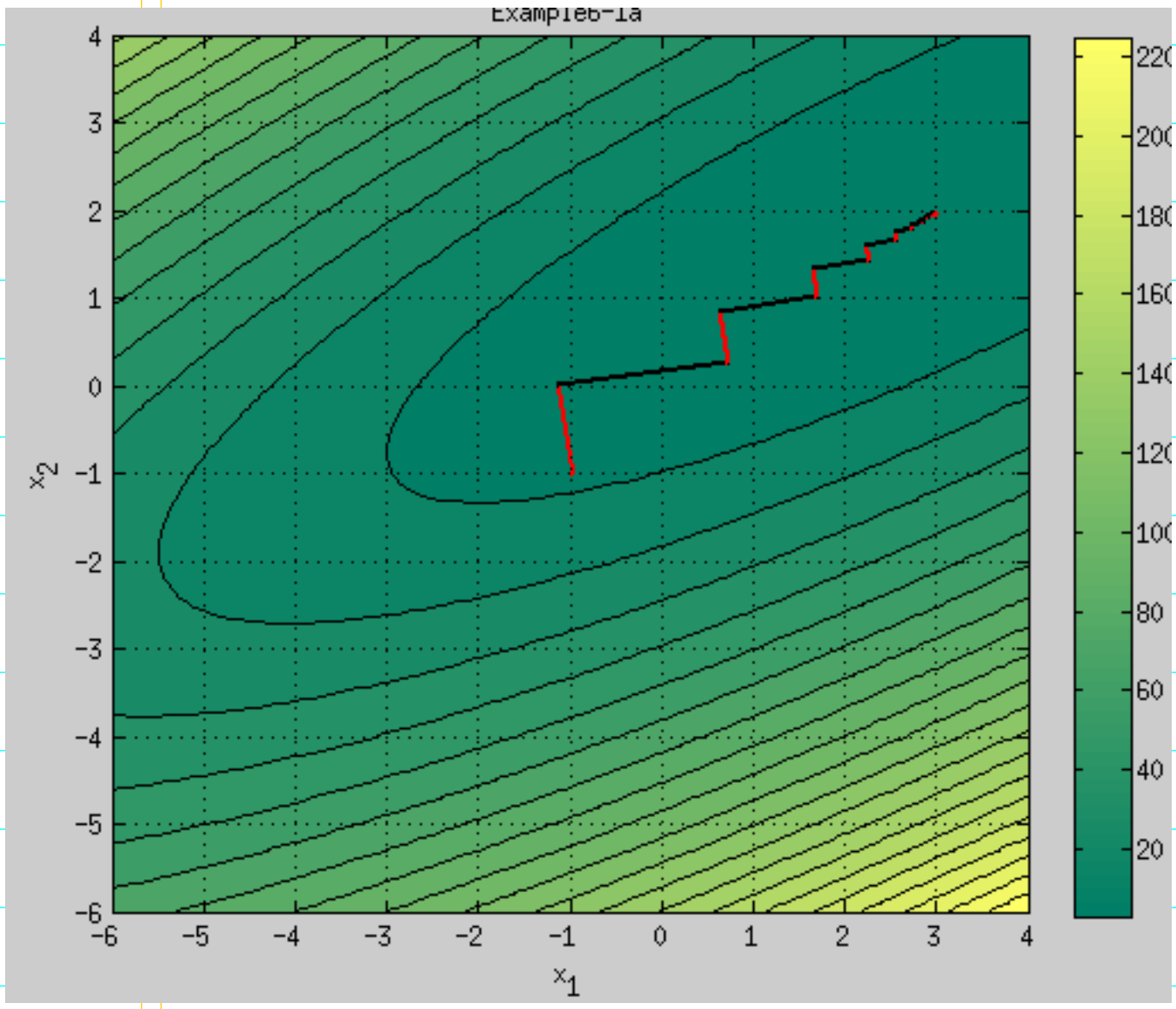


iterations	x(1)	x(2)	f
1	0.50000	0.50000	5.31250
2	0.50000	0.50000	5.31250
3	0.54900	0.49244	5.30870
.	.	.	.
.	.	.	.
16	1.41137	1.08211	3.88738
17	1.41137	1.08211	3.88738
18	1.35789	1.13970	3.86378
19	1.34387	1.16389	3.86066
20	1.34387	1.16389	3.86066

Total cpu time (s)= 1.2000

Steepest (Gradient) descent for example @

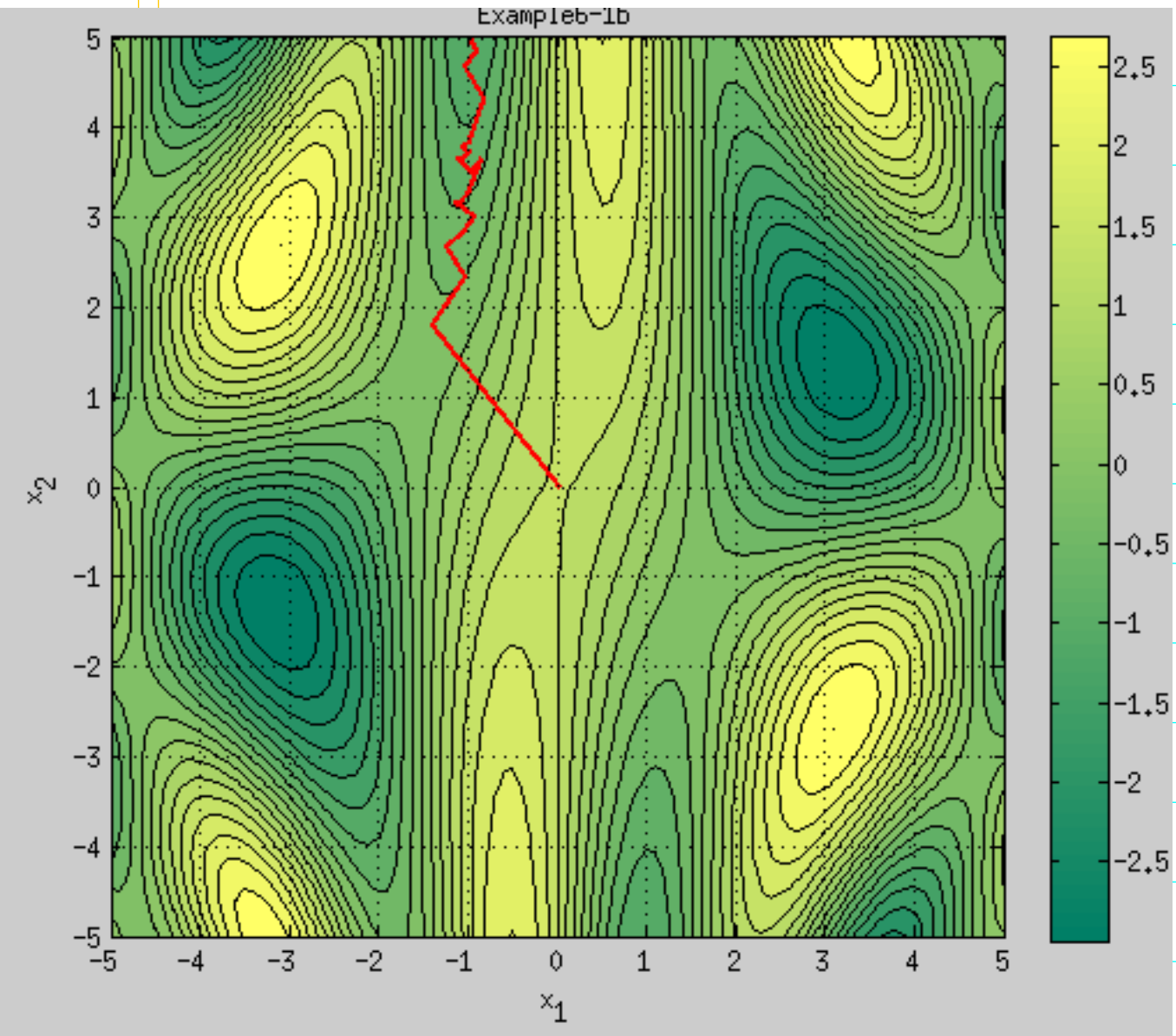
http://www.cse.iitb.ac.in/~CS709/notes/code/unconstrainedOpt/SteepestDescent_Example6_1a.m



iter	x(1)	x(2)	f	KT Cond	alpha	DesignChange
1	-1.00000	-1.00000	12.25000	5.720e+01	0.000e+00	0.000e+00
2	-1.13892	0.04044	8.27888	5.852e+00	1.388e-01	1.050e+00
3	0.71031	0.28496	6.02139	1.860e+01	7.711e-01	1.865e+00
4	0.63158	0.87888	4.72852	1.916e+00	1.389e-01	5.991e-01
.	.	.	.			
.	.	.	.			
26	2.97379	1.98913	3.00022	4.790e-04	3.109e-01	5.808e-03
27	2.97690	1.98727	3.00018	1.530e-04	1.656e-01	3.624e-03
28	2.98083	1.99298	3.00012	4.944e-04	5.601e-01	6.929e-03
29	2.98298	1.99098	3.00009	5.434e-05	1.319e-01	2.932e-03
30	2.99501	1.99957	3.00002	2.972e-04	2.006e+00	1.479e-02

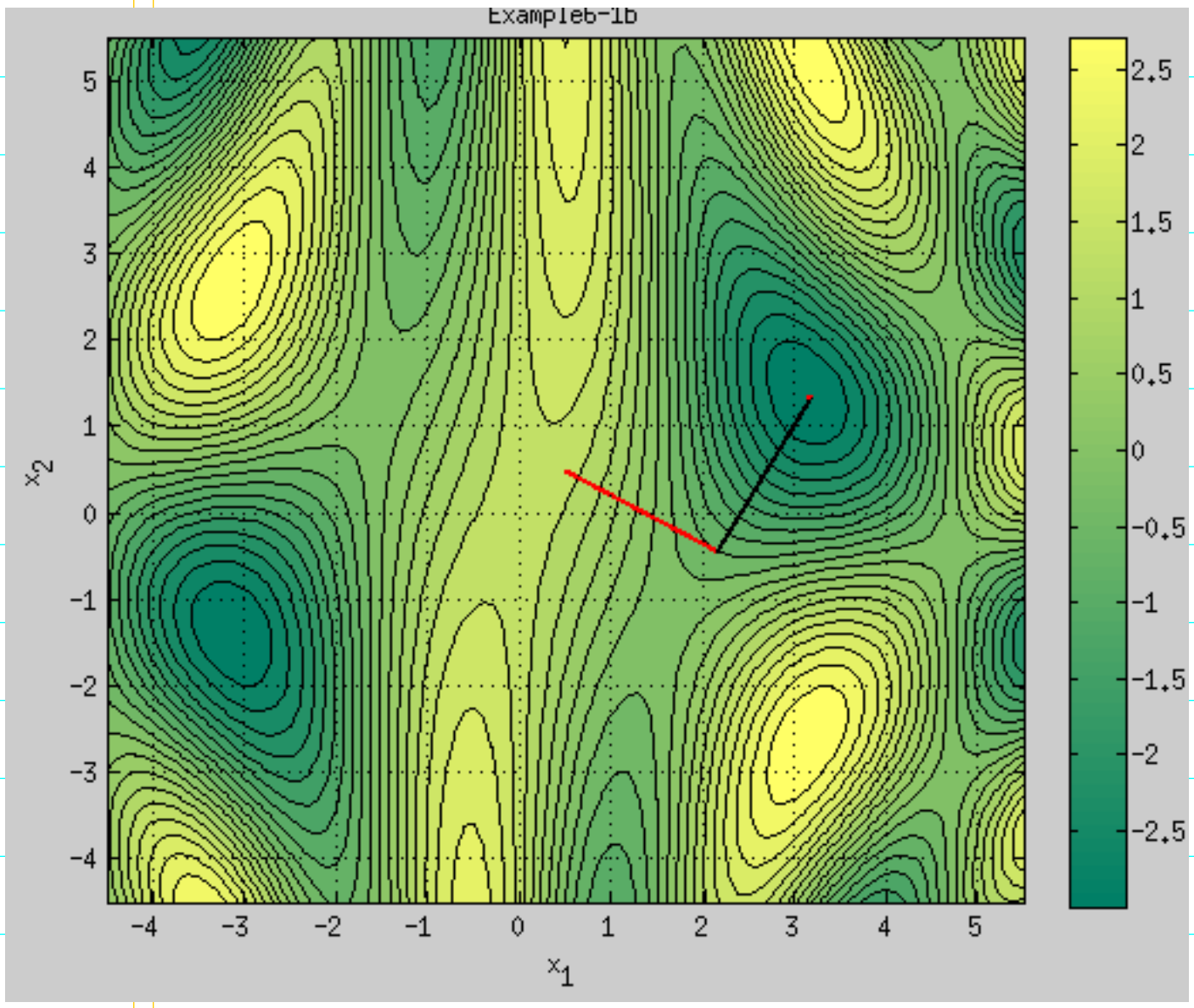
Total cpu time (s)= 2.1300

Try for example (b) at http://www.cse.iitb.ac.in/~CS709/notes/code/unconstrainedOpt/RandomWalk_Example6_1b.m



Steepest descent on problem (b) at

http://www.cse.iitb.ac.in/~CS709/notes/code/unconstrainedOpt/SteepestDescent_Example6_1b.m

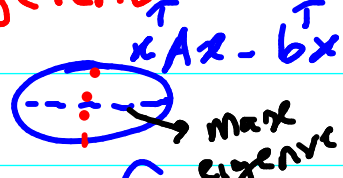


iter	x(1)	x(2)	f	KT Cond	alpha	DesignChange
1	0.50000	0.50000	1.40405	3.171e-01	0.000e+00	0.000e+00
2	2.14931	-0.43793	-0.42913	9.572e-01	3.369e+00	1.897e+00
3	3.16033	1.34155	-2.99930	2.820e-03	2.092e+00	2.047e+00
4	3.14088	1.35182	-2.99987	5.403e-04	4.143e-01	2.200e-02
5	3.14474	1.36002	-2.99998	9.840e-05	3.897e-01	9.058e-03
6	3.14180	1.36087	-2.99999	1.269e-05	3.087e-01	3.062e-03
7	3.14154	1.36339	-3.00000	3.100e-06	7.111e-01	2.533e-03
8	3.14154	1.36339	-3.00000	3.100e-06	0.000e+00	0.000e+00

Total cpu time (s)= 1.0000

For (c) http://www.cse.iitb.ac.in/~CS709/notes/code/unconstrainedOpt/SteepestDescent_Example6_2.m

It can be proved that if f is a strongly convex function and continuously doubly differentiable in $\text{int}(\text{dmn } f)$ s.t



$$mI \leq \nabla^2 f(x) \leq MI \qquad \forall x \in \text{dmn } f$$

(a) ⇒ sublevel sets in dmn(f) & therefore dmn(f) are bounded & therefore max eigenvalue of $\nabla^2 f$ is bounded above

then $\forall x, y, \in \text{dmn}(f)$

Already proved

$$f(y) \geq f(x) + \langle \nabla f(x), y-x \rangle + \frac{m}{2}\|y-x\|^2 \quad (a)$$

and

$$f(y) \leq f(x) + \langle \nabla f(x), y-x \rangle + \frac{M}{2}\|y-x\|^2 \quad (b)$$

Proof for (b) is analogous to proof for (a)

Now recall the Backtracking ray search algo:

1. $k=0$, make an initial guess x^0
2. First you choose Δx^k → $\Delta x^k = -\nabla f(x^k)$ for gradient descent
   Until $f(x^k - t^k \Delta x^k) \leq f(x^k) - \alpha t^k \nabla^T f(x^k) \Delta x^k$.
   search for t^k (such as $t^k = 1$, $t^k = \beta t^k$ until condition is met]
3. $x^{k+1} = x^k + t^k \Delta x^k$ & $k = k+1$
4. Until $\|\nabla f(x^k)\| < \epsilon$

Rough understanding of rate of convergence for gradient descent for strongly convex function under backtracking ray search with Wolfe's first condition

(A)

(1) Strong convexity: $f(y) \le f(x) + \langle \nabla f(x), y-x \rangle + \frac{M}{2}\|y-x\|^2$

& $f(y) \ge f(x) + \langle \nabla f(x), y-x \rangle + \frac{m}{2}\|y-x\|^2$

$mI \le \nabla^2 f(x) \le MI \quad \forall\ x \in \text{dmn } f$

(2) $\boxed{\min(1, \beta/M) \le t_k \le \frac{1}{M}}$ and $\Delta x^k = -\nabla f(x^k)$

(3) $f(x^{k+1}) = f\left(x^k - t_k \nabla f(x^k)\right)$

$\le f(x^k) - t_k \|\nabla f(x^k)\|_2^2 + \frac{M}{2} t_k^2 \|\nabla f(x^k)\|_2^2$

$\le f(x^k) - \frac{t_k}{2}\|\nabla f(x^k)\|_2^2 \quad (\because M t_k \le 1)$

(4) $f(x^{k+1}) \le f(x^k) - \alpha \min(1, \beta/M)\|\nabla f(x)\|^2$ (wolfe condition 1)

$\Rightarrow f(x^{k+1}) - f(x^*) \le f(x^k) - f(x^*) - \alpha \min(1, \beta/M)\|\nabla f(x)\|^2$

$\le \underbrace{(1 - 2m\alpha(1, \beta/M))}_{c} \left(f(x^k) - f(x^*)\right)$

Recall from strong convexity $\|\nabla f(x)\|_2^2 \ge 2m(f(x) - f(x^*))$

Combining the two ⇒

$$f(x^{k+1}) - f(x^*) \leq c\left(f(x^k) - f(x^*)\right) \leq c^{k+1}\left(f(x^0) - f(x^*)\right)$$

Linear convergence with $c = \left(1 - 2m\alpha \min(1, \beta/M)\right)$

Rate of Convergence definitions

Given a sequence $\{s^k\}$ that converges to s^*, that is: $\lim_{k\to\infty} \|s^k - s^*\| = 0$, we say that the order of convergence is p, where $p \in \mathbb{R}$ if

$$0 < \lim_{k\to\infty} \frac{\|s^{k+1} - s^*\|}{\|s^k - s^*\|^p} = M < \infty$$

and M is called the rate of convergence

if $s^k = f(x^k)$ & $s^* = f(x^*)$ then $\|\cdot\|$ is simply absolute value $|\cdot|$

Special cases:

(a) Superlinear convergence: $p = 1 \Rightarrow M = 0$ } Eg: $s^k = 1 + k^{-k}$

(b) Sublinear convergence: $p = 1 \Rightarrow M = 1$ } Eg: $s^k = \frac{1}{k+1}$ $(s^* = 0)$

(c) Linear convergence: $p=1 \Rightarrow M \in (0,1)$

Eg: $s^k = 1+(0.5)^k$, $M = \boxed{0.5}$ $(s^* = 1)$ [Relate to grad desc]

(d) Quadratic convergence (a kind of superlinear convergence)

$p=1 \Rightarrow M = \boxed{0}$ and $p=2 \Rightarrow M < \infty$

Eg: $1+(0.5)^{2^k}$, $M = \boxed{1}$ $(s^* = 1)$

(e) Cubic convergence (a kind of superlinear convergence)

$p=1 \Rightarrow M = 0$ and $p=3 \Rightarrow M < \infty$

(H/W: Find an s_k with cubic convergence)

Rate of convergence (for steepest descent) without needing strong convexity

Claim: Suppose $f: \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable, and the iterates generated by steepest descent with exact line search converge to a point x^* at which $\nabla^2 f(x^*)$ is positive definite. Let r be a scalar s.t

$$r \in \left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}, 1 \right)$$

Claim of linear convergence w/o strong convexity

where $0 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ are the eigenvalues of $\nabla^2 f(x^*)$. Then, for all sufficiently large k:

$$f(x^{k+1}) - f(x^*) \leq r^2 \left[f(x^k) - f(x^*) \right]$$

Proof: Hint... for most convergence proofs involving the Hessian, first try proving convergence for the objective

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

$$\nabla^2 f(x) = A$$

for $A \in S_n^{++}$ and write $\lim_{k \to \infty} \frac{\| f(x^{k+1}) - f(x^*) \|}{\| f(x^k) - f(x^*) \|}$ (you will realise its value $= r^2$)

Next do this for $\hat{f}(x) = f(x^0) + \nabla f(x^0)^T (x - x^0) + \frac{1}{2} (x - x^0)^T \nabla^2 f(x^0) (x - x^0)$

Q: How to prove convergence/rate of convergence results for other choices of steepest descent (ie $\underset{v}{\operatorname{argmin}}\{v^T \nabla f(x) \mid \|v\|_p = 1\}$ for $p \neq 2$) such as for (B), (C) or (D)?

Ans: Use the fact that any norm $\|x\|_p$ can be bounded by $\|x\|_2$ i.e $\exists\ r \& \tilde{r} \in (0,1]$ s.t.

$$\|x\|_p \geq r\|x\|_2 \quad \text{and} \quad \|x\|_q \geq \tilde{r}\|x\|_2$$

where $\frac{1}{p} + \frac{1}{q} = 1$ (i.e $\|\cdot\|_q$ is dual norm of $\|\cdot\|_p$)

Eg: If $p = 1$ we can see that:

$$\|x\|_1 \geq \|x\|_2 \quad \text{(equality iff only one } x_i \neq 0\text{)}$$

$$\|x\|_\infty \geq \frac{1}{\sqrt{n}}\|x\|_2 \quad \text{(equality iff } x_1 = x_2 = \cdots = x_n\text{)}$$

Conjugate gradient method

Generalisation of Coordinate descent method

Other perspectives and further details can be found at

http://www.cse.iitb.ac.in/~CS709/notes/BasicsOfConvexOptimization.pdf

pages 313–324 (sections 4.5.7 and 4.5.8)

Idea behind conjugate gradient method:

Consider (as usual) the quadratic optimisation problem

$$\min_{x} \frac{1}{2}x^TAx - x^Tb + c \qquad (1)$$

where $A \in S^{n\times n}_{++}$ ie $A \succ 0$ and $A \in \mathbb{R}^{n\times n}$

(1) Instead of choosing Δx^{k+1} as orthogonal to Δx^k (as in coordinate descent), suppose we chose directions $\{d_0, d_2 \ldots d_k, d_{k+1} \ldots d_{n-1}\}$ that are "conjugate" wrt A

i.e

$$d_i^T A d_j = 0 \quad \forall\, i \neq j$$

(2) Show that $d_0, d_2 \ldots d_{n-1}$ are linearly independent

Proof by contradiction i.e suppose $\exists \alpha_1, \alpha_2 \ldots \alpha_n$ not all 0 s.t $\sum_{i=0}^{n-1} \alpha_{i+1} d_i = 0$

(3) Since $d_1, d_2 \ldots d_n$ are linearly independent, solution x^* to (1) can be expressed as

$$x^n = x^* = \sum_{i=0}^{n-1} \alpha_{i+1} d_i \rightarrow (2)$$

From first order necessary (and sufficient) conditions for minimum of (1) at x^*

$$Ax^* = b \rightarrow (3)$$

Combining (2) and (3) and premultiplying by d_k^T:

$$\sum_{i=0}^{n-1} \alpha_{i+1} d_k^T A d_i = d_k^T b \qquad (\because d_j^T A d_i = 0 \quad \forall i \neq j)$$

$$\Rightarrow \boxed{\alpha_{k+1} = \frac{d_k^T b}{d_k^T A d_k}} \quad \text{or} \quad \boxed{\alpha_k = \frac{d_{k-1}^T b}{d_{k-1}^T A d_{k-1}}}$$

(4) Define: $x^k = \sum_{i=0}^{k-1} \alpha_{i+1} d_i$ and $x^0 = 0$

Then: $\boxed{x^k = x^{k-1} + \alpha_k d_{k-1}}$.

(5) Also let $r_k = b - Ax^k$ (i.e. $-\nabla f(x^k)$ and residual)

$\therefore r_0 = b$ & $r_k = b - A(x^{k-1} + \alpha_k d_{k-1})$

r_{k-1}

$\therefore r_k = r_{k-1} - \alpha_k A d_{k-1}$

⑥ Now we would like to obtain d_k from $d_0 \dots d_{k-1}$ → $\{d_0 .. d_{k-1}\}$ is called a Krylov subspace $K_n(r_0, k-1)$

Let $\boxed{d_0 = r_0}$ and $\boxed{d_k = r_k + \beta_k d_{k-1}}$ → Remove from residual r_k components covered by $d_0 .. d_{k-1}$

Then $\beta_k = \boxed{-\frac{d_{k-1}^T A r_k}{d_{k-1}^T A d_{k-1}}} = \frac{r_k^T r_k}{r_{k-1}^T r_{k-1}}$

verify: Purpose is to minimize matrix multiplications

We summarise the Conjugate gradient algo for the quadratic objective from page 321 of http://www.cse.iitb.ac.in/~CS709/notes/BasicsOfConvexOptimization.pdf
(Correctness & convergence by construction of algo)

$\mathbf{x}_0 = \mathbf{0}$, $\mathbf{r}_0 = \mathbf{b}$, $\mathbf{d}_0 = \mathbf{r}_0$, $k = 1$.
repeat
1. $\alpha_k = \frac{\mathbf{r}_{k-1}^T \mathbf{r}_{k-1}}{\mathbf{d}_{k-1}^T A \mathbf{d}_{k-1}}$. //Step length for next update. This corresponds to the entry $H_{k,k}$.
2. $\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha_k \mathbf{d}_{k-1}$.
3. $\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_k A \mathbf{d}_{k-1}$. //New residual obtained using $\mathbf{r}_k - \mathbf{r}_{k-1} = -A(\mathbf{x}_k - \mathbf{x}_{k-1})$.
4. $\beta_k = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{r}_{k-1}^T \mathbf{r}_{k-1}}$. //Improvement over previous step. This corresponds to the entry $H_{k,k+1}$.
5. $\mathbf{d}_k = \mathbf{r}_k + \beta_k \mathbf{d}_{k-1}$. //The next search direction, which should be orthogonal to the search direction just used.

$k = k + 1$.
until $\beta_k < \theta$.

Figure 4.52: The conjugate gradient algorithm for solving $A\mathbf{x} = \mathbf{b}$ or equivalently, for minimizing $E(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{x}^T \mathbf{b}$.

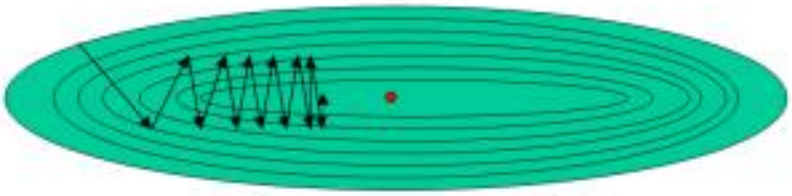


Figure 4.53: Illustration of the steepest descent technique on level curves of the function $E(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{x}^T\mathbf{b}$.

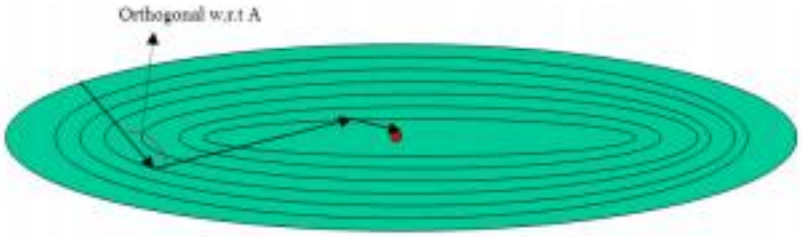


Figure 4.54: Illustration of the conjugate gradient technique on level curves of the function $E(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{x}^T\mathbf{b}$.

FLETCHER REEVES

Conjugate gradient for non-quadratic $f(x)$: Replace (a) $r_0 = b$ with $g_0 = \nabla f(x^{(0)})$

(b) $r_k = b - Ax_k = r_{k-1} - \alpha_k A d_{k-1}$ with $g^{(k)} = \nabla f(x^{(k)})$ (c) α_k obtained using line search (d) $d_k = -g^{(k)} + \beta_k d^{(k-1)}$

Select $\mathbf{x}^{(0)}$, Let $f_0 = f(\mathbf{x}^{(0)})$, $\mathbf{g}_0 = \nabla f(\mathbf{x}^{(0)})$, $\mathbf{d}^{(0)} = -\nabla \mathbf{g}_0$, $k = 1$.

repeat

1. Compute α_k by line search.
2. Set $\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} + \alpha_k \mathbf{d}^{(k-1)}$.
3. Evaluate $\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)})$.
4. $\beta_k = \dfrac{(\mathbf{g}^{(k)})^T \mathbf{g}^{(k)}}{(\mathbf{g}^{(k-1)})^T \mathbf{g}^{(k-1)}}$. → several variants possible here (see notes)
5. $\mathbf{d}_k = -\mathbf{g}^{(k)} + \beta_k \mathbf{d}^{(k-1)}$.

$k = k + 1$.

until $\dfrac{\|\mathbf{g}^{(k)}\|}{\|\mathbf{g}^{(0)}\|} < \theta$ OR $k > maxIter$.

Figure 4.55: The conjugate gradient algorithm for optimizing nonlinear convex function f.

VARIANTS

Variants of the Fletcher-Reeves method use different choices of the parameter β_k. An important variant, proposed by Polak and Ribiere, defines β_k as

$$\beta_k^{PR} = \frac{\left(\mathbf{g}^{(k)}\right)^T \left(\mathbf{g}^{(k)} - \mathbf{g}^{(k-1)}\right)}{\left(\mathbf{g}^{(k)}\right)^T \mathbf{g}^{(k)}}$$

The Fletcher-Reeves method converges if the starting point is sufficiently close to the desired minimum. However, convergence of the Polak-Ribiere method can be guaranteed by choosing

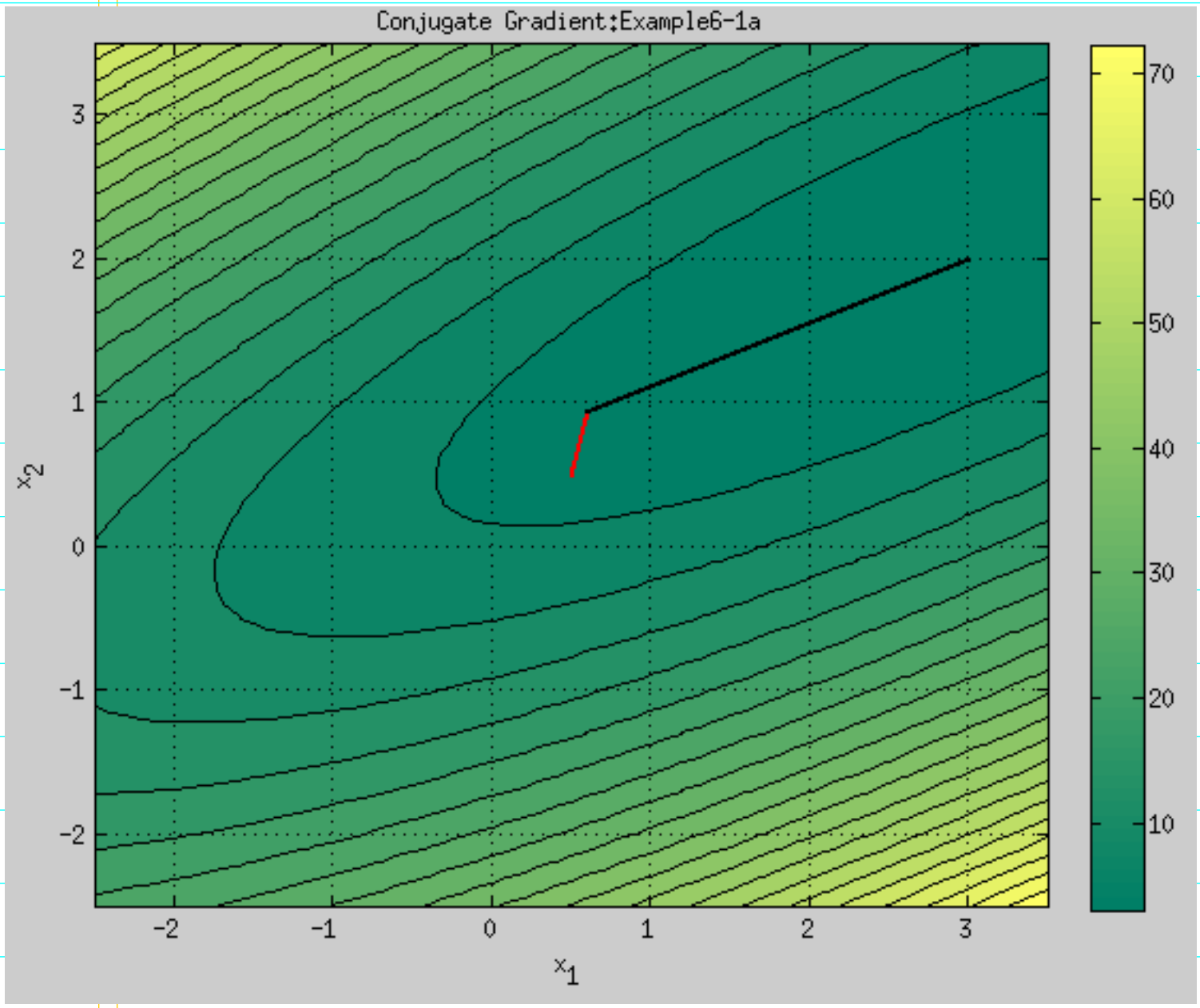
$$\beta_k = max\left\{\beta_k^{PR}, 0\right\}$$

Using this value is equivalent to restarting[34] conjugate gradient if $\beta_k^{PR} < 0$. In practice, the Polak-Ribiere method converges much more quickly than the Fletcher-Reeves method. It is generally required to restart the conjugate gradient method after every n iterations, in order to get back conjugacy, *etc.*

Conjugate gradient belongs to the larger class of iterative methods, in particular, KRYLOV methods (see pages 314 onwards of http://www.cse.iitb.ac.in/~CS709/notes/BasicsOfConvexOptimization.pdf). Among all possible methods, whose first k steps are restricted to the Krylov subspace $K_n(r_0, k)$, conjugate gradient does the best job of minimizing the distance to optimal soln x^* after k steps...

Let us run conjugate gradient on example ⓐ

http://www.cse.iitb.ac.in/~CS709/notes/code/unconstrainedOpt/ConjugateGradient_Example6_1a.m



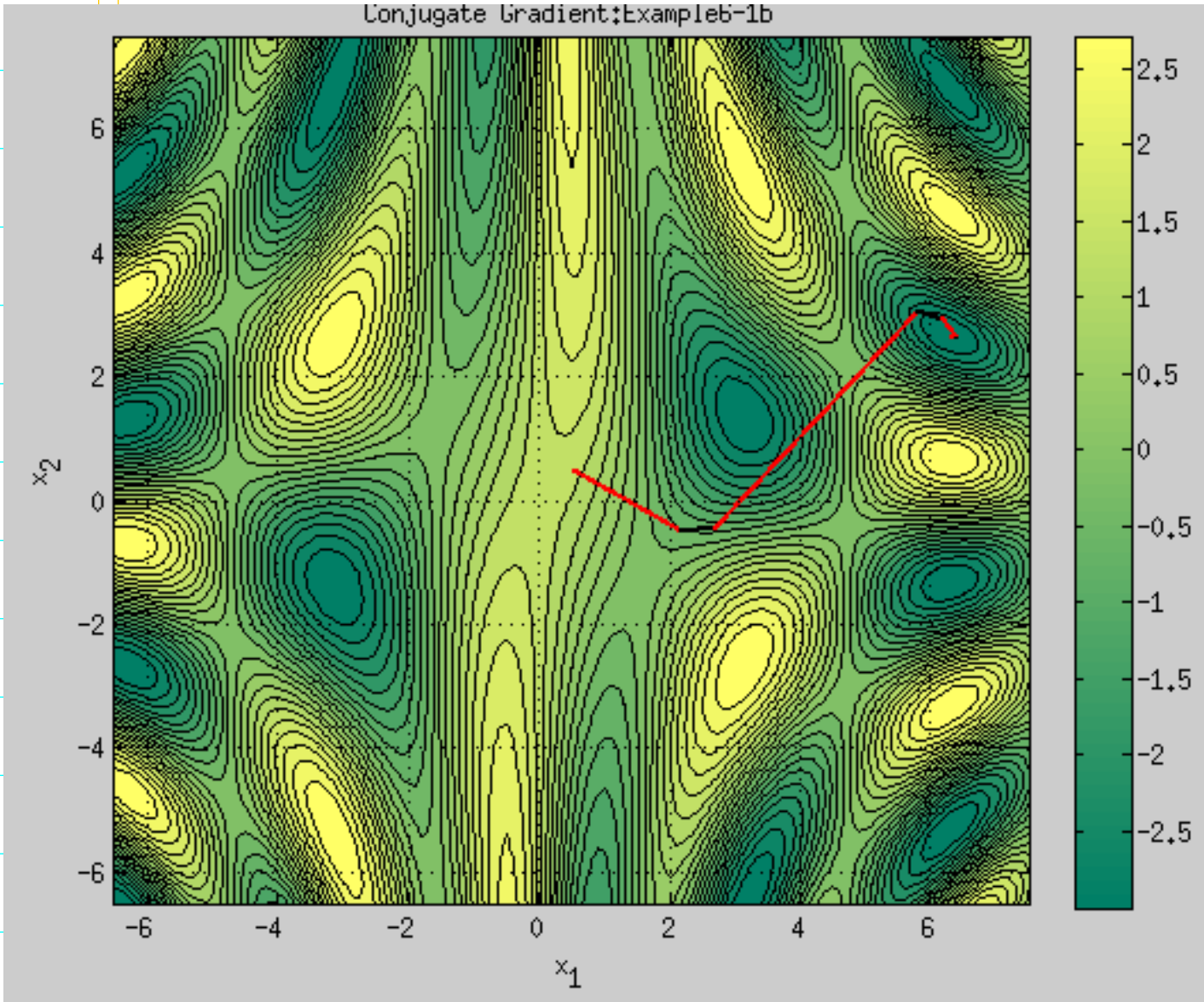
- The design vector,function value, KT condition etc. during the iterations

iter	x(1)	x(2)	f	KT Cond	alpha	DesignChange
1	0.50000	0.50000	5.31250	2.490e-01	0.000e+00	0.000e+00
2	0.59958	0.94835	4.78321	2.842e+00	1.996e-01	4.593e-01
3	2.99956	1.99807	3.00001	9.810e-05	1.255e+00	2.620e+00
4	2.99865	1.99931	3.00000	7.959e-06	1.549e-01	1.533e-03
5	2.99865	1.99931	3.00000	7.959e-06	0.000e+00	0.000e+00

Total cpu time (s)= 0.8900 → Contrast with 2.1 s reqd using gradient descent (page 23!)

How about conjugate gradient on example (b)?

http://www.cse.iitb.ac.in/~CS709/notes/code/unconstrainedOpt/ConjugateGradient_Example6_1b.m



- The design vector,function value, KT condition etc. during the iterations

iter	x(1)	x(2)	f	KT Cond	alpha	DesignChange
1	0.50000	0.50000	1.40405	2.396e-01	0.000e+00	0.000e+00
2	2.14931	-0.43793	-0.42913	9.572e-01	3.369e+00	1.897e+00
3	2.68002	-0.43511	-0.55592	3.100e+00	2.707e-01	5.307e-01
4	5.78797	3.06439	-2.57496	1.601e+00	1.978e+00	4.680e+00
5	6.14284	2.97857	-2.77816	2.773e+00	2.548e-01	3.651e-01
6	6.35664	2.68067	-2.98852	1.537e-01	1.437e-01	3.667e-01
7	6.32658	2.65096	-2.99661	1.662e-02	1.064e-01	4.226e-02
8	6.28185	2.67247	-2.99964	5.790e-03	3.628e-01	4.963e-02
9	6.28122	2.68248	-2.99999	2.615e-05	1.149e-01	1.003e-02
10	6.28269	2.68133	-3.00000	1.510e-06	3.801e-01	1.866e-03
11	6.28337	2.68150	-3.00000	1.386e-05	5.031e-01	7.022e-04
12	6.28337	2.68150	-3.00000	1.386e-05	0.000e+00	0.000e+00

Total cpu time (s)= 1.6100 → slower than gradient descent (1 sec) on page 25

H/w: You earlier applied a wobbly gradient descent to the Rosenbrock fn $f(x_1,x_2) = 100(x_2 - x_1^2)^2 + (1-x_1)^2$
Now try applying Conjugate gradient to this fn
[You can try modifying code at http://www.cse.iitb.ac.in/~CS709/notes/code/unconstrainedOpt/ConjugateGradient_Example6_1a.m to observe the trajectory graphically]

Rate of convergence: • For Quadratic objective convergence rate is linear (convergence rate is function of condition # ie K(A).

• If A has k distinct eigenvalues, conjugate gradient converges in k steps to optimum of Quadratic objective

Newton step

scaling & rotating gradient

$$\Delta x_{\text{nt}} = -\nabla^2 f(x)^{-1} \nabla f(x)$$

typically step length not too imp

interpretations

- $x + \Delta x_{\text{nt}}$ minimizes second order approximation

$$\widehat{f}(x+v) = f(x) + \nabla f(x)^T v + \frac{1}{2} v^T \nabla^2 f(x) v$$

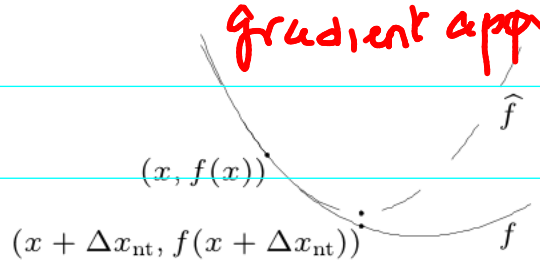
fn approx

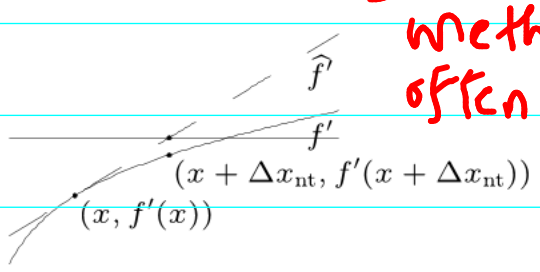
- $x + \Delta x_{\text{nt}}$ solves linearized optimality condition

$$\nabla f(x+v) \approx \nabla \widehat{f}(x+v) = \nabla f(x) + \nabla^2 f(x) v = 0$$

gradient approx

} Quasi Newton methods often use this

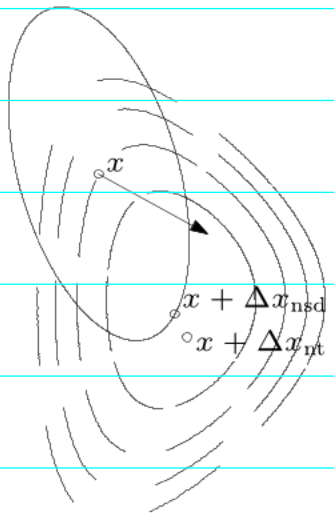




- $\Delta x_{\rm nt}$ is steepest descent direction at x in local Hessian norm

$$\|u\|_{\nabla^2 f(x)} = \left(u^T \nabla^2 f(x) u\right)^{1/2}$$

Recall case ② of steepest descent with $P = \nabla^2 f(x^k)$



dashed lines are contour lines of f; ellipse is $\{x + v \mid v^T \nabla^2 f(x) v = 1\}$

arrow shows $-\nabla f(x)$

Newton decrement

$\lambda(x) = \left(\nabla^T f(x) \Delta x\right)^{1/2}$

$$\lambda(x) = \left(\nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x)\right)^{1/2}$$

Recall that $\nabla^T f(x) \Delta x < 0$ is necessary for descent in case of convex fn

a measure of the proximity of x to $x^\star$

properties

- gives an estimate of $f(x) - p^\star$, using quadratic approximation $\widehat{f}$:

$$f(x) - \inf_y \widehat{f}(y) = \frac{1}{2}\lambda(x)^2$$

- equal to the norm of the Newton step in the quadratic Hessian norm

$$\lambda(x) = \left(\Delta x_{\rm nt} \nabla^2 f(x) \Delta x_{\rm nt}\right)^{1/2}$$

- directional derivative in the Newton direction: $\nabla f(x)^T \Delta x_{\rm nt} = -\lambda(x)^2$
- affine invariant (unlike $\|\nabla f(x)\|_2$)

Newton's method

---

given a starting point $x \in \mathbf{dom}\, f$, tolerance $\epsilon > 0$.
repeat

1. *Compute the Newton step and decrement.*
   $\Delta x_{\rm nt} := -\nabla^2 f(x)^{-1}\nabla f(x)$; $\quad \lambda^2 := \nabla f(x)^T \nabla^2 f(x)^{-1}\nabla f(x)$.
2. *Stopping criterion.* **quit** if $\lambda^2/2 \leq \epsilon$.
3. *Line search.* Choose step size t by backtracking line search.
4. *Update.* $x := x + t\Delta x_{\rm nt}$.

In general descent algo we used $\|\nabla f(x)\| < \epsilon$

affine invariant, *i.e.*, independent of linear changes of coordinates:

Newton iterates for $\tilde{f}(y) = f(Ty)$ with starting point $y^{(0)} = T^{-1}x^{(0)}$ are

$$y^{(k)} = T^{-1}x^{(k)}$$

Will Newton's algo converge? (Recall from page 13)

Claim: Consider the descent algo in figure 4.45 with $x^{(k+1)} = x^{(k)} + t^{(k)}\Delta x^{(k)}$ where $\Delta x^{(k)}$ is a descent direction and $t^{(k)}$ satisfies the Wolfe conditions. Suppose f is bounded below in $\mathbb{R}^n$ and that f is continuously differentiable in an open set N containing the level set $\{x : f(x) \leq f(x^{(0)})\}$ where $x^{(0)}$ is the starting point if the iteration. Also, let $f \in F_L$. Then

$$\sum_{k=0}^{\infty} \frac{\left(\nabla^T f(x^{(k)})\Delta x^{(k)}\right)^2}{\|\Delta x^{(k)}\|^2} \text{ is finite ie } < \infty$$

we saw that (vii) is

sufficient to ensure that $\lim_{k\to\infty} \|\nabla f(x^k)\| = 0$ (page 16)

$$\frac{-\nabla^T f(x^{(k)}) \Delta x^{(k)}}{\|\Delta x^{(k)}\| \, \|\nabla f(x^{(k)})\|} \geq r > 0 \quad \forall\, k \qquad \text{(vii)}$$

Claim: For Newton (and Quasi-Newton) algorithms that have

$$\Delta x^k = -(B_k)^{-1} \nabla f(x^k)$$

$B_k = \nabla^2 f(x^k)$ for Newton's method

where $B_k \in S_n^{++}$

if the condition number of B_k (for all k) is bounded i.e

$$\|B_k\| \|B_k^{-1}\| \leq M \quad \forall k$$

then

$$-\frac{\nabla^T f(x^k) \Delta x^k}{\|\nabla f(x^k)\| \|\Delta x^k\|} \geq \frac{1}{M}$$

Aside: Condition num of B is relative error in soln x of $Bx = C$, divided by the relative error in C.

Let $B(x+\delta) = C + \epsilon$ & $Bx = C$. Then $\delta = B^{-1}\epsilon$

$\Rightarrow \|\delta\| \leq \|B^{-1}\| \|\epsilon\| \Rightarrow \frac{\|\delta\|}{\|x\|} \leq \frac{\|B^{-1}\| \|\epsilon\|}{\|x\|}$. → (b)

$Bx = C$ (a) ⇓ $\|B\|\|x\| \geq \|c\|$

By definition of induced matrix norm: $\frac{\|Bx\|}{\|x\|} \leq \|B\|$ → (c)

Combining (a), (b) and (c): $\frac{\|\delta\|}{\|x\|} \leq \underbrace{\|B\|\|B^{-1}\|}\, \frac{\|\epsilon\|}{\|c\|} = \text{cond}(B) \frac{\|\epsilon\|}{\|b\|}$

PROOF

For any $B \in S_n^{++}$, $\det(B) > 0$ & $\therefore B$ is invertible

$$\Rightarrow \|x\| = \|B^{-1}Bx\| \leq \|B^{-1}\| \|Bx\|$$

$$\Rightarrow \|Bx\| \geq \frac{\|x\|}{\|B^{-1}\|}$$

Since $B \in S_n^{++}$, $B^{1/2}$ & $B^{-1/2}$ exist (see previous discussions) and it can be proved that

$$\|B^{1/2}\| = \|B\|^{1/2} \quad \& \quad \|B^{-1/2}\| = \|B^{-1}\|^{1/2}$$

$$\Rightarrow \frac{\nabla^T f(x) \Delta x}{\|\nabla f(x)\| \|\Delta x\|} = \frac{\Delta x^T B \Delta x}{\|B\Delta x\| \|\Delta x\|}$$

$$\geq \frac{\Delta x^T B \Delta x}{\|B\| \|\Delta x\|^2} \quad (\because \|B\Delta x\| \leq \|B\| \|\Delta x\|)$$

$$= \frac{\Delta x^T B^{1/2} B^{1/2} \Delta x}{\|B\| \|\Delta x\|^2}$$

$$= \frac{\|B^{1/2}\Delta x\|^2}{\|B\| \|\Delta x\|^2}$$

$$\geq \frac{\|\Delta x\|^2}{\|B^{-1/2}\|^2 \|B\| \|\Delta x\|^2} \quad \left(\because \|B^{1/2}\Delta x\| \geq \frac{\|\Delta x\|}{\|B^{-1/2}\|}\right)$$

$$= \frac{1}{\|B^{-1}\| \|B\|} \geq \frac{1}{M}$$

Hence proved!

Newton's method

---

given a starting point $x \in \mathbf{dom}\, f$, tolerance $\epsilon > 0$.
repeat

1. *Compute the Newton step and decrement.*
   $\Delta x_{\rm nt} := -\nabla^2 f(x)^{-1}\nabla f(x)$; $\quad \lambda^2 := \nabla f(x)^T \nabla^2 f(x)^{-1}\nabla f(x)$.
2. *Stopping criterion.* **quit** if $\lambda^2/2 \leq \epsilon$.
3. *Line search.* Choose step size t by backtracking line search.
4. *Update.* $x := x + t\Delta x_{\rm nt}$.

---

affine invariant, *i.e.*, independent of linear changes of coordinates:

Newton iterates for $\tilde{f}(y) = f(Ty)$ with starting point $y^{(0)} = T^{-1}x^{(0)}$ are

$$y^{(k)} = T^{-1}x^{(k)}$$

Classical convergence analysis

assumptions

- f strongly convex on S with constant m
- $\nabla^2 f$ is Lipschitz continuous on S, with constant $L > 0$:

$$\|\nabla^2 f(x) - \nabla^2 f(y)\|_2 \leq L\|x - y\|_2$$

(L measures how well f can be approximated by a quadratic function)

outline: there exist constants $\eta \in (0, m^2/L)$, $\gamma > 0$ such that

- if $\|\nabla f(x)\|_2 \geq \eta$, then $f(x^{(k+1)}) - f(x^{(k)}) \leq -\gamma$
- if $\|\nabla f(x)\|_2 < \eta$, then

$$\frac{L}{2m^2}\|\nabla f(x^{(k+1)})\|_2 \leq \left(\frac{L}{2m^2}\|\nabla f(x^{(k)})\|_2\right)^2$$

damped Newton phase ($\|\nabla f(x)\|_2 \geq \eta$)

- most iterations require backtracking steps
- function value decreases by at least γ
- if $p^\star > -\infty$, this phase ends after at most $(f(x^{(0)}) - p^\star)/\gamma$ iterations

since for x close to x^*, $\nabla^2 f(x) \approx \nabla^2 f(x^*)$

quadratically convergent phase ($\|\nabla f(x)\|_2 < \eta$)

- all iterations use step size $t = 1$
- $\|\nabla f(x)\|_2$ converges to zero quadratically: if $\|\nabla f(x^{(k)})\|_2 < \eta$, then

$$\frac{L}{2m^2}\|\nabla f(x^l)\|_2 \leq \left(\frac{L}{2m^2}\|\nabla f(x^k)\|_2\right)^{2^{l-k}} \leq \left(\frac{1}{2}\right)^{2^{l-k}}, \qquad l \geq k$$

conclusion: number of iterations until $f(x) - p^\star \leq \epsilon$ is bounded above by

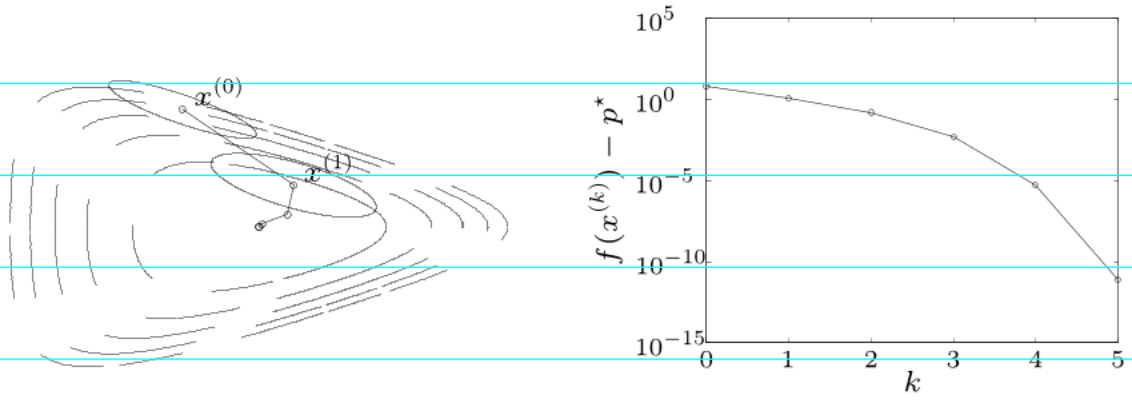
$$\frac{f(x^{(0)}) - p^\star}{\gamma} + \log_2 \log_2(\epsilon_0/\epsilon)$$

- γ, ϵ_0 are constants that depend on m, L, $x^{(0)}$
- second term is small (of the order of 6) and almost constant for practical purposes
- in practice, constants m, L (hence γ, ϵ_0) are usually unknown
- provides qualitative insight in convergence properties (*i.e.*, explains two algorithm phases)

H/W: we know that if $f(x)$ is quadratic, Newton should take 1 iteration. Find corresponding $\epsilon_0, \epsilon, \gamma$ etc

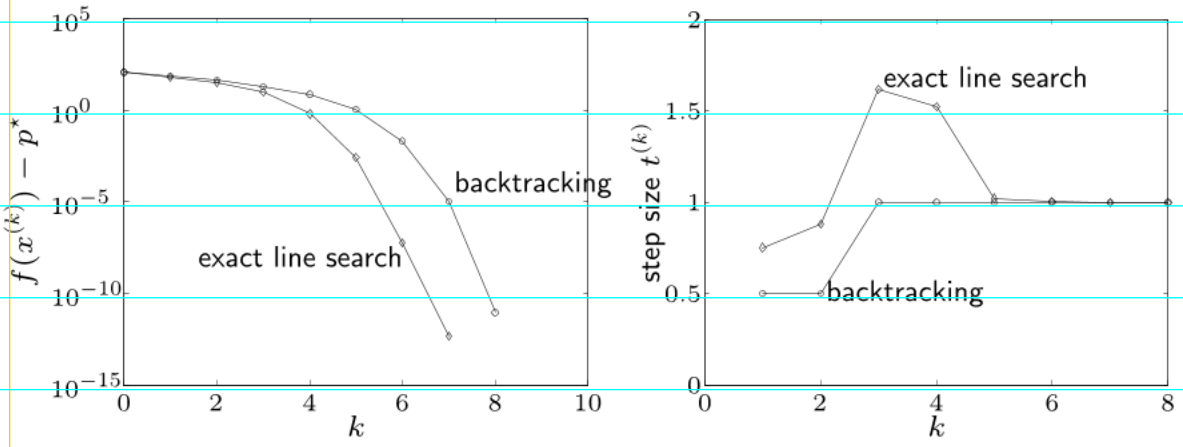
Examples

example in $\mathbf{R}^2$ (page 10–9)



- backtracking parameters $\alpha = 0.1$, $\beta = 0.7$
- converges in only 5 steps
- quadratic local convergence

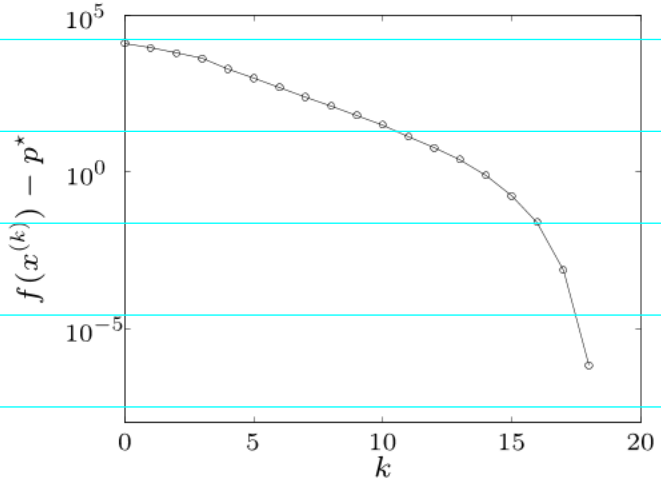
example in $\mathbf{R}^{100}$ (page 10–10)



- backtracking parameters $\alpha = 0.01$, $\beta = 0.5$
- backtracking line search almost as fast as exact l.s. (and much simpler)
- clearly shows two phases in algorithm

example in $\mathbf{R}^{10000}$ (with sparse a_i)

$$f(x) = -\sum_{i=1}^{10000} \log(1 - x_i^2) - \sum_{i=1}^{100000} \log(b_i - a_i^T x)$$



- backtracking parameters $\alpha = 0.01$, $\beta = 0.5$.
- performance similar as for small examples

Newton method

special cases

1) Gauss Newton
   - Objective that can be decomposed into a sum over (training) examples

$$f(x) = \frac{1}{2}\sum_{i=1}^{I} (y_i - r(t_i, x))^2$$

Approximations for general objective functions

(Alleviate expensive (a) Hessian computation (b) inverse computation)

1) Broyden-Fletcher-Goldfarb-Shanno (BFGS)
   - Uses linear algebra to iteratively compute
   - Preceded by DFP algo

2) Levenberg-Marquardt

Interpolation of Gauss Newton and gradient descent algos

3) Self-concordant fns

→ Special class of convex fns closed under affine transform etc. Eg: Negative logarithm, negative log determinant, linear, quadratic etc. . . .

→ Convergence analysis does not depend on any unknown constants

Estimate $B^{(k)}$ of $(\nabla^2 f(x^k))^{-1}$ ensuring

(a) → that $B^{(k)} \in S_n^{++}$

(b) → that quadratic approx fn using $B^{(k+1)}$ gives correct gradient at x^{k+1}

(c) → that $B^{(k+1)}$ is not too far from $B^{(k)}$

(d) → Reducing complexity from $O(n^{2.7})$ to $O(n^2)$ per iteration

2) Limited memory BFGS (LBFGS) Limits rank of B^k

10/23

Quasi Newton methods

1) Gauss Newton Approximation {Self Reading}

(section 4.5.4 of http://www.cse.iitb.ac.in/~CS709/notes/BasicsOfConvexOptimization.pdf)

The Gauss Newton method decomposes the objective function (typically for a regression problem) as a composition of two functions[27] $f = l \circ \mathbf{m}$; (i) the vector valued model or regression function $\mathbf{m} : \Re^n \rightarrow \Re^p$ and (ii) the scalar-valued loss (such as the sum squared difference between predicted outputs and target outputs) function l. For example, if m_i is $y_i - r(\mathbf{t}_i, \mathbf{x})$, for parameter vector $\mathbf{x} \in \Re^n$ and input instances $(y_i, \mathbf{t}_i)$ for $i = 1, 2, \ldots, p$, the function f can be written as

$$f(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^{p} (y_i - r(\mathbf{t}_i, \mathbf{x}))^2$$

An example of the function r is the linear regression function $r(\mathbf{t}_i, \mathbf{x}) = \mathbf{x}^T\mathbf{t}_i$. Logistic regression poses an example objective function, which involves a cross-entropy loss.

$$f(\mathbf{x}) = -\sum_{i=1}^{p} \left(y_i \log\left(\sigma(\mathbf{x}^T\mathbf{t}_i)\right) + (1-y_i)\log\left(\sigma(-\mathbf{x}^T\mathbf{t}_i)\right)\right)$$

where $\sigma(k) = \frac{1}{1+e^{-k}}$ is the logistic function.

The Hessian $\nabla^2 f(\mathbf{x})$ can be expressed using a matrix version of the chain rule, as

$$\nabla^2 f(\mathbf{x}) = \underbrace{J_{\mathbf{m}}(\mathbf{x})^T \nabla^2 l(\mathbf{m}) J_{\mathbf{m}}(\mathbf{x})}_{G_f(\mathbf{x})} + \sum_{i=1}^{p} \nabla^2 m_i(\mathbf{x})(\nabla l(\mathbf{m}))_i$$

→ From chain rule

where $J_{\mathbf{m}}$ is the jacobian[28] of the vector valued function $\mathbf{m}$. It can be shown that if $\nabla^2 l(\mathbf{m}) \succeq 0$, then $G_f(\mathbf{x}) \succeq 0$. The term $G_f(\mathbf{x})$ is called the Gauss-Newton approximation of the Hessian $\nabla^2 f(\mathbf{x})$. In many situtations, $G_f(\mathbf{x})$ is the dominant part of $\nabla^2 f(\mathbf{x})$ and the approximation is therefore reasonable.

Q: If x is point of minimum, $\nabla^2 f(x) = \boxed{G_f(x)}$

The (approximate) Newton update rule will be:

$$\Delta\mathbf{x} = -(G_f(\mathbf{x}))^{-1}\nabla f(\mathbf{x}) = -(G_f(\mathbf{x}))^{-1} J_{\mathbf{m}}^T(\mathbf{x})\nabla l(\mathbf{m})$$

where we use the fact that $(\nabla f(\mathbf{x}))_i = \sum_{k=1}^{p} \frac{\partial l}{\partial m_k}\frac{\partial m_k}{\partial x_i}$, since the gradient of a composite function is a product of the jacobians.

② Levenberg-Marquardt

→ Aim: To control the large step size taken by Newton method and bring more stability

Section 4.5.5 of http://www.cse.iitb.ac.in/~CS709/notes/BasicsOfConvexOptimization.pdf

$$\Delta \mathbf{x} = -\left(G_f(\mathbf{x}) + \lambda \operatorname{diag}(G_f)\right)^{-1} J_{\mathbf{m}}^T(\mathbf{x}) \nabla l(\mathbf{m})$$

where G_f is the Gauss-Newton approximation to $\nabla^2 f(\mathbf{x})$ and is assumed to be positive semi-definite. This method is one of the work-horses of modern optimization. The parameter $\lambda \geq 0$ adaptively controlled, limits steps to an elliptical model-trust region[29]. This is achieved by adding λ to the smallest eigenvalues of G_f, thus restricting all eigenvalues of the matrix to be above λ so that the elliptical region has diagonals of shorter length that inversely vary as the eigenvalues (*c.f.* page 3.11.3). While this method fixes the stability issues in Newtons method, it still requires the $O(n^3)$ time required for matrix inversion.

Self-concordance

shortcomings of classical convergence analysis

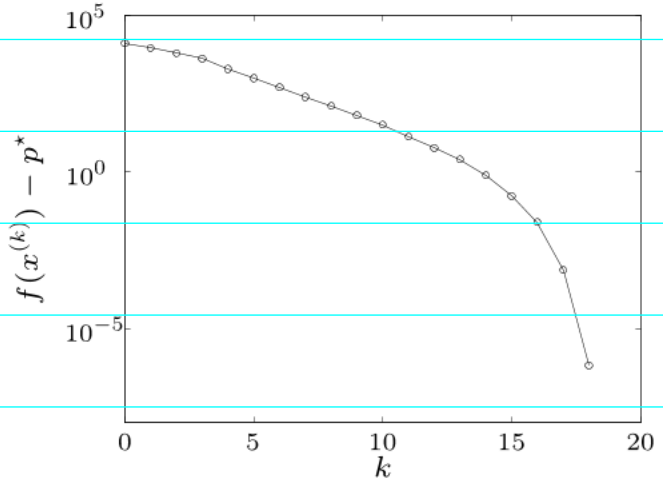
- depends on unknown constants (m, L, $\dots$)
- bound is not affinely invariant, although Newton's method is

convergence analysis via self-concordance (Nesterov and Nemirovski)

- does not depend on any unknown constants
- gives affine-invariant bound
- applies to special class of convex functions ('self-concordant' functions)
- developed to analyze polynomial-time interior-point methods for convex optimization

example in $\mathbf{R}^{10000}$ (with sparse a_i)

$$f(x) = -\sum_{i=1}^{10000} \log(1 - x_i^2) - \sum_{i=1}^{100000} \log(b_i - a_i^T x)$$



- backtracking parameters $\alpha = 0.01$, $\beta = 0.5$.
- performance similar as for small examples

Self-concordance

shortcomings of classical convergence analysis

- depends on unknown constants (m, L, . . .)
- bound is not affinely invariant, although Newton's method is

convergence analysis via self-concordance (Nesterov and Nemirovski)

- does not depend on any unknown constants
- gives affine-invariant bound
- applies to special class of convex functions ('self-concordant' functions)
- developed to analyze polynomial-time interior-point methods for convex optimization

Self-concordant functions

definition

- $f : \mathbf{R} \to \mathbf{R}$ is self-concordant if $|f'''(x)| \leq 2f''(x)^{3/2}$ for all $x \in \mathbf{dom}\, f$
- $f : \mathbf{R}^n \to \mathbf{R}$ is self-concordant if $g(t) = f(x + tv)$ is self-concordant for all $x \in \mathbf{dom}\, f$, $v \in \mathbf{R}^n$

examples on R

- linear and quadratic functions
- negative logarithm $f(x) = -\log x$
- negative entropy plus negative logarithm: $f(x) = x \log x - \log x$

affine invariance: if $f : \mathbf{R} \to \mathbf{R}$ is s.c., then $\tilde{f}(y) = f(ay + b)$ is s.c.:

$$\tilde{f}'''(y) = a^3 f'''(ay + b), \qquad \tilde{f}''(y) = a^2 f''(ay + b)$$

Self-concordant calculus

properties

- preserved under positive scaling $\alpha \geq 1$, and sum
- preserved under composition with affine function
- if g is convex with $\mathbf{dom}\, g = \mathbf{R}_{++}$ and $|g'''(x)| \leq 3g''(x)/x$ then

$$f(x) = \log(-g(x)) - \log x$$

  is self-concordant

examples: properties can be used to show that the following are s.c.

- $f(x) = -\sum_{i=1}^m \log(b_i - a_i^T x)$ on $\{x \mid a_i^T x < b_i,\ i = 1, \ldots, m\}$
- $f(X) = -\log\det X$ on $\mathbf{S}_{++}^n$
- $f(x) = -\log(y^2 - x^T x)$ on $\{(x, y) \mid \|x\|_2 < y\}$

Convergence analysis for self-concordant functions

summary: there exist constants $\eta \in (0, 1/4]$, $\gamma > 0$ such that

- if $\lambda(x) > \eta$, then
$$f(x^{(k+1)}) - f(x^{(k)}) \leq -\gamma$$
- if $\lambda(x) \leq \eta$, then
$$2\lambda(x^{(k+1)}) \leq \left(2\lambda(x^{(k)})\right)^2$$

(η and γ only depend on backtracking parameters α, β)

Damped phase

Quadratic convergence

complexity bound: number of Newton iterations bounded by

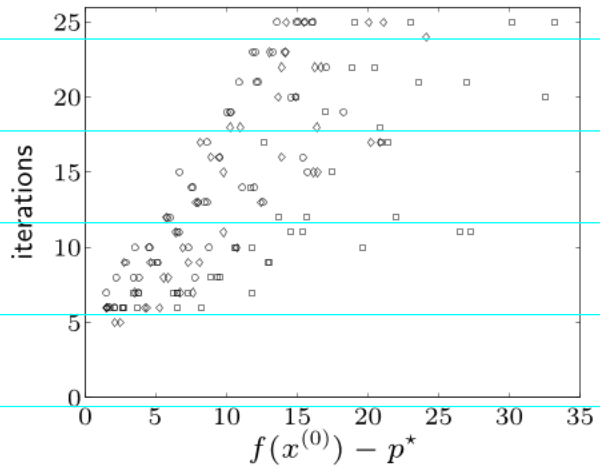
$$\frac{f(x^{(0)}) - p^\star}{\gamma} + \log_2 \log_2(1/\epsilon)$$

for $\alpha = 0.1$, $\beta = 0.8$, $\epsilon = 10^{-10}$, bound evaluates to $375(f(x^{(0)}) - p^\star) + 6$

numerical example: 150 randomly generated instances of

$$\text{minimize} \quad f(x) = -\sum_{i=1}^m \log(b_i - a_i^T x)$$

○: $m = 100$, $n = 50$
□: $m = 1000$, $n = 500$
◇: $m = 1000$, $n = 50$



- number of iterations much smaller than $375(f(x^{(0)}) - p^\star) + 6$
- ~~bound of the form $c(f(x^{(0)}) - p^\star) + 6$ with smaller c (empirically) valid~~

Implementation

main effort in each iteration: evaluate derivatives and solve Newton system

$$H\Delta x = g$$

where $H = \nabla^2 f(x)$, $g = -\nabla f(x)$

via Cholesky factorization

$$H = LL^T, \qquad \Delta x_{\rm nt} = L^{-T}L^{-1}g, \qquad \lambda(x) = \|L^{-1}g\|_2$$

- cost $(1/3)n^3$ flops for unstructured system
- cost $\ll (1/3)n^3$ if H sparse, banded

example of dense Newton system with structure

$$f(x) = \sum_{i=1}^n \psi_i(x_i) + \psi_0(Ax + b), \qquad H = D + A^T H_0 A$$

- assume $A \in \mathbf{R}^{p\times n}$, dense, with $p \ll n$
- D diagonal with diagonal elements $\psi_i''(x_i)$; $H_0 = \nabla^2\psi_0(Ax + b)$

method 1: form H, solve via dense Cholesky factorization: (cost $(1/3)n^3$)

method 2 (page 9–15): factor $H_0 = L_0L_0^T$; write Newton system as

$$D\Delta x + A^T L_0 w = -g, \qquad L_0^T A\Delta x - w = 0$$

eliminate Δx from first equation; compute w and Δx from

$$(I + L_0^T AD^{-1}A^T L_0)w = -L_0^T AD^{-1}g, \qquad D\Delta x = -g - A^T L_0 w$$

cost: $2p^2n$ (dominated by computation of $L_0^T AD^{-1}A^T L_0$)

BFGS Quasi Newton method. [Section 4.5.6 of http://www.cse.iitb.ac.in/~CS709/notes/BasicsOfConvexOptimization.pdf]

FIND estimate $B^{(k)}$ of $(\nabla^2 f(x^k))^{-1}$ ensuring

(a) → that $B^{(k)} \in S_n^{++}$

(b) → that quadratic approx fn using $B^{(k+1)}$ gives correct gradient at x^k & at $x^{(k+1)}$

(c) → that $B^{(k+1)}$ is not too far from $B^{(k)}$

(d) → Reducing complexity from $O(n^{2.7})$ to $O(n^2)$ per iteration

$\Delta x^{(k)} = -B^{(k)} \nabla f(x^{(k)})$ with $B^{(k)} \succ 0$. That is, $\Delta x^{(k)}$ is the minimizer of the convex quadratic model

$$Q^{(k)}(p) = f(x^{(k)}) + \nabla^T f(x^{(k)}) p + \frac{1}{2} p^T \left(B^{(k)}\right)^{-1} p$$

$x^{(k+1)} = x^{(k)} + t^{(k)} \Delta x^{(k)}$, where $t^{(k)}$ is obtained by line search.

Questions:

Why (a)?

How to ensure (b)?

How to ensure (c)?

How does (d) happen?

Idea: $\nabla^2 f(x^{k+1}) \approx \nabla^2 f(x^k) + \nabla^2 f_u$ (Updates)

$\Rightarrow B^{(k+1)} = B^{(k)} + B_u^k$ (translates to)

$B^k \approx [\nabla^2 f(x^k)]^{-1}$

Why (a): Recall from page 40 that if:

$B^k \in S_n^{++} \; \forall k$ and

$\|B^k\| \|(B^k)^{-1}\| \leq M \; \forall k$

then all (Quasi) Newton methods will converge

How to ensure (b)?

The gradient of the function $Q^{(k+1)} = f(\mathbf{x}^{(k+1)}) + \nabla^T f(\mathbf{x}^{(k+1)})\mathbf{p} + \frac{1}{2}\mathbf{p}^T \left(B^{(k+1)}\right)^{-1} \mathbf{p}$ at $\mathbf{p} = \mathbf{0}$ and $\mathbf{p} = -t^{(k)}\Delta\mathbf{x}^{(k)}$ agrees with gradient of f at $\mathbf{x}^{(k+1)}$ and $\mathbf{x}^{(k)}$ respectively. While the former condition is naturally satisfied, the latter need to be imposed. This quasi-Newton condition yields

Latest 2 iterates

$$\left(B^{(k+1)}\right)^{-1}\left(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right) = \nabla f(\mathbf{x}^{(k+1)}) - \nabla f(\mathbf{x}^{(k)}). \quad (*)$$

This equation is called the *secant equation*.

General idea: (additive updates)

$$B^{(k+1)} = B^k + B_u^k \rightarrow \text{update matrix}$$

s.t $B_u^k = R^k + S^k$

Each is a rank-1 symmetric matrix satisfying (*)

$$R^k = a(v^k)(v^k)^T$$

$$S^k = b(u^k)(u^k)^T$$

$b = 0 \Rightarrow$ rank 1 Quasi Newton

$b \neq 0 \Rightarrow$ rank 2 Quasi Newton

where $a, b \in \mathbb{R}$

v^k, u^k are vectors

Different choices of $R^{(k)}$ & $S^{(k)}$ give different Rank-2 Quasi Newton Methods: Eg: DFP & BFGS

We will discuss BFGS

How to ensure (C)?

Finally, among all symmetric matrices satisfying the secant equation, $B^{(k+1)}$ is closest to the current matrix $B^{(k)}$ in some norm. Different matrix norms give rise to different quasi-Newton methods. In particular, when the norm chosen is the Frobenius norm, we get the following BGFS update rule

$$B^{(k+1)} = B^{(k)} + R^{(k)} + S^{(k)}$$

where,

$$R^{(k)} = \frac{\Delta \mathbf{x}^{(k)} \left(\Delta \mathbf{x}^{(k)}\right)^T}{\left(\Delta \mathbf{x}^{(k)}\right)^T \Delta \mathbf{g}^{(k)}} - \frac{B^{(k)} \Delta \mathbf{g}^{(k)} \left(\Delta \mathbf{g}^{(k)}\right)^T \left(B^{(k)}\right)^T}{\left(\Delta \mathbf{g}^{(k)}\right)^T B^{(k)} \Delta \mathbf{g}^{(k)}}$$

and

$$S^{(k)} = \mathbf{u} \left(\Delta \mathbf{x}^{(k)}\right)^T B^{(k)} \Delta \mathbf{x}^{(k)} \mathbf{u}^T$$

with

$$\mathbf{u} = \frac{\Delta \mathbf{x}^{(k)}}{\left(\Delta \mathbf{x}^{(k)}\right)^T \Delta \mathbf{g}^{(k)}} - \frac{B^{(k)} \Delta \mathbf{g}^{(k)}}{\left(\Delta \mathbf{g}^{(k)}\right)^T B^{(k)} \Delta \mathbf{g}^{(k)}}$$

We have made use of the Sherman Morrison formula that determines how updates to a matrix relate to the updates to the inverse of the matrix.

What about (d)?

Find a starting point $\mathbf{x}^{(0)} \in \mathcal{D}$ and an approximate $B^{(0)}$ (which could be I). → amounts to initial step = gradient descent step.
Select an appropriate tolerance $\epsilon > 0$.
repeat
1. Set $\Delta\mathbf{x}^{(k)} = -B^{(k)}\nabla f(\mathbf{x}^{(k)})$.
2. Let $\lambda^2 = \nabla^T f(\mathbf{x}^{(k)})B^{(k)}\nabla f(\mathbf{x}^{(k)})$.
3. If $\frac{\lambda^2}{2} \leq \epsilon$, **quit**.
4. Set step size $t^{(k)} = 1$.
5. Obtain $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + t^{(k)}\Delta\mathbf{x}^{(k)}$.
6. Compute $\Delta\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k+1)}) - \nabla f(\mathbf{x}^{(k)})$.
7. Compute $R^{(k)}$ and $S^{(k)}$. → $O(n^2)$
8. Compute $B^{(k+1)} = B^{(k)} + R^{(k)} + S^{(k)}$. → $O(n^2)$
6. Set $k = k + 1$.

until

Figure 4.50: The BFGS method.

- How about Limited memory BFGS? (LBFGS)

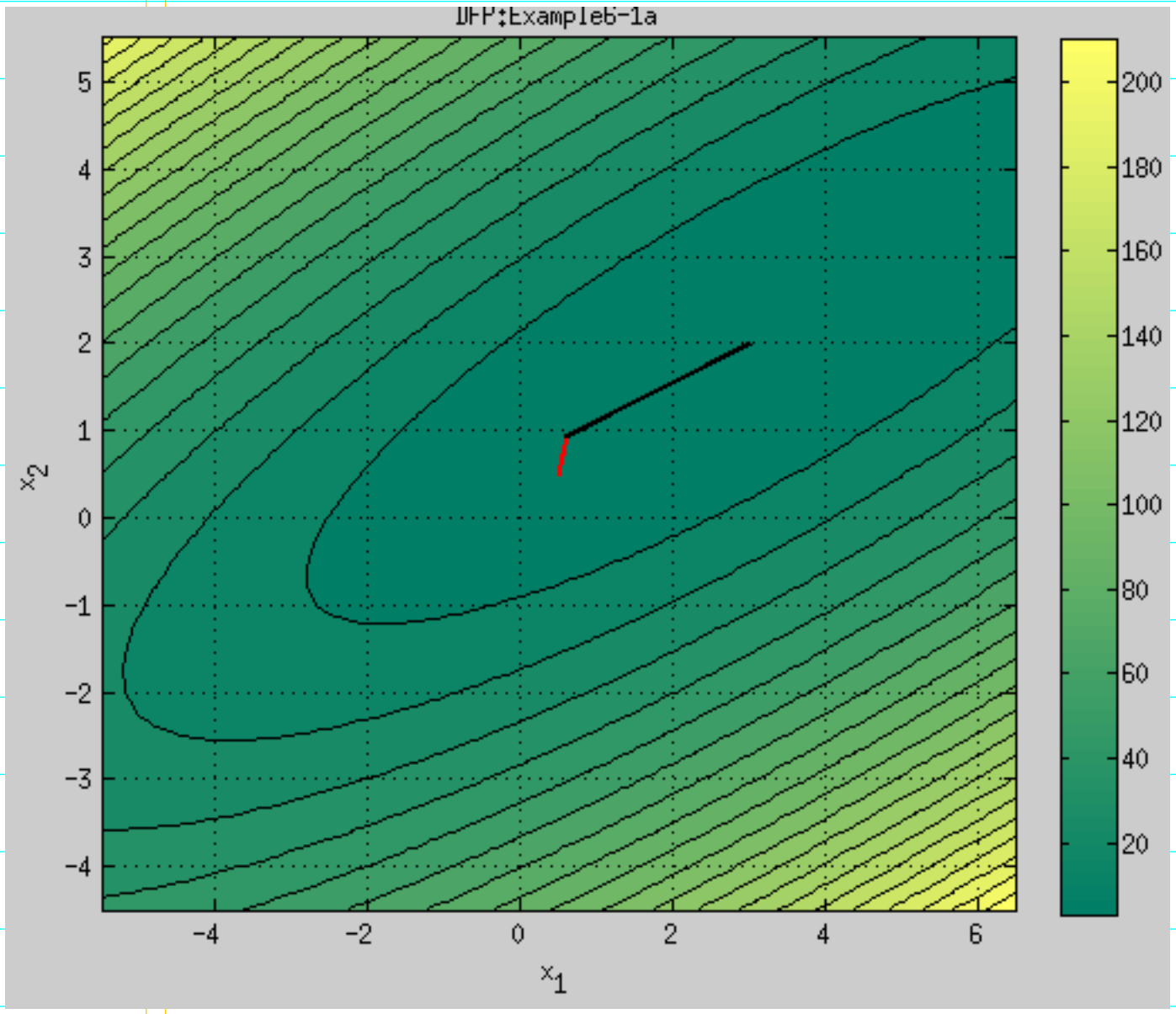
LBFGS employs a limited-memory quasi-Newton approximation that does not require much storage or computation. It limites the rank of the inverse of the hessian to some number $\gamma \in \Re$ so that only $n\gamma$ numbers have to be stored instead of n^2 numbers. For general non-convex problems, LBFGS may fail when the initial geometry (in the form of $B^{(0)}$) has been placed very close to a saddle point. Also, LBFGS is very sensitive to line search.

Recently, L-BFGS has been observed to be the most effective parameter estimation method for Maximum Entropy model, much better than improved iterative scaling (IIS) and generalized iterative scaling (GIS).

- DFP (Davidon Fletcher Powell) is a variation of BFGS and preceeded it

Quasi Newton method DFP on example ⓐ

http://www.cse.iitb.ac.in/~CS709/notes/code/unconstrainedOpt/DFP_Example6_1a.m



iter	x(1)	x(2)	f	KT Cond	alpha	DesignChange
1	0.50000	0.50000	5.31250	5.297e+00	0.000e+00	0.000e+00
2	0.59958	0.94835	4.78321	2.842e+00	1.996e-01	4.593e-01
3	2.99971	1.99873	3.00000	3.538e-05	1.929e+00	2.620e+00
4	2.99836	1.99833	3.00000	1.462e-05	3.571e-01	1.404e-03
5	2.99836	1.99833	3.00000	1.462e-05	0.000e+00	0.000e+00

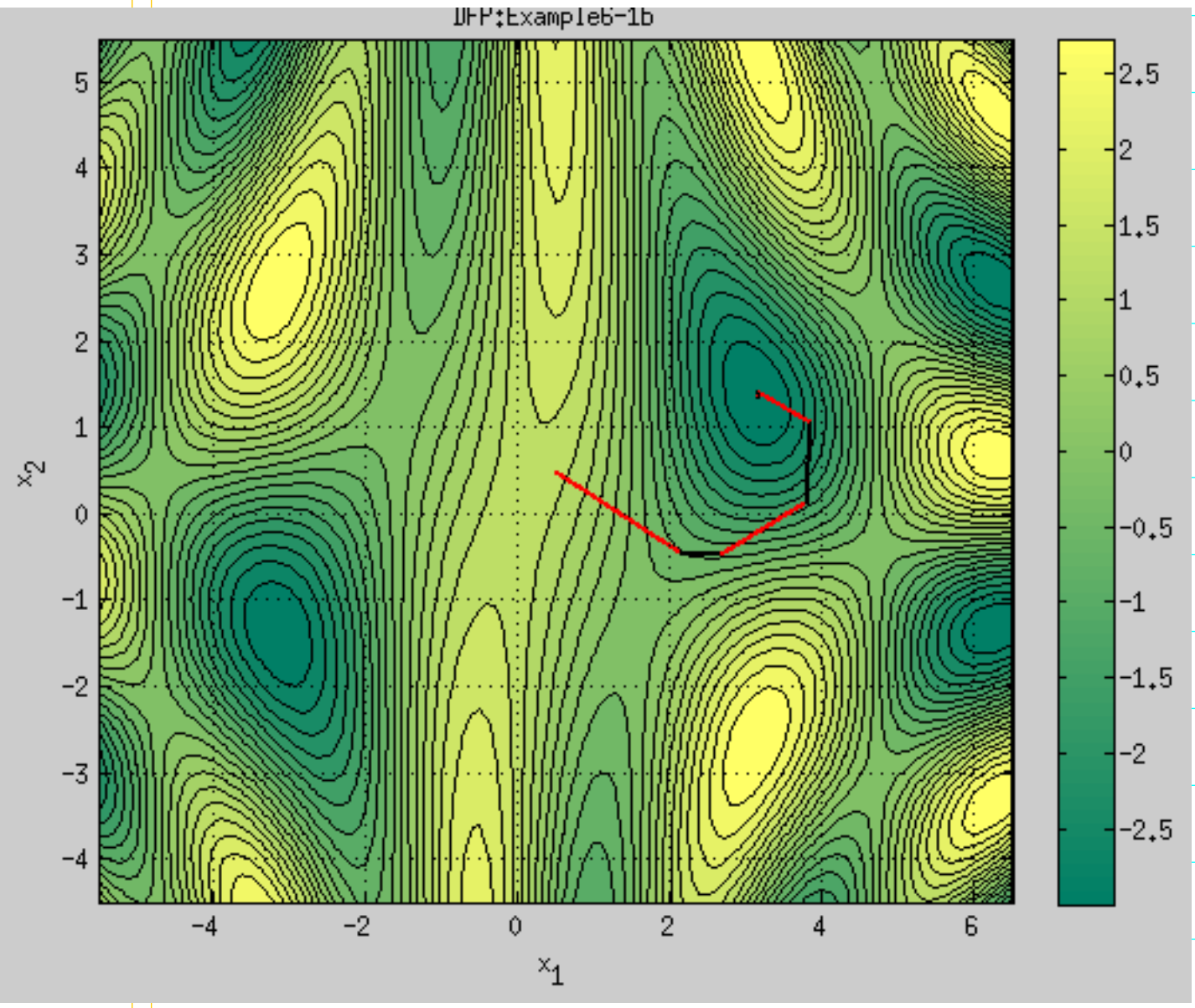
The Final Metric

1.6250	0.7500
0.7500	0.5000

Total cpu time (s)= 0.8100

Quasi Newton method DFP on example (b)

http://www.cse.iitb.ac.in/~CS709/notes/code/unconstrainedOpt/ConjugateGradient_Example6_1b.m



iter	x(1)	x(2)	f	KT Cond	alpha	DesignChange
1	0.50000	0.50000	1.40405	3.171e-01	0.000e+00	0.000e+00
2	2.14931	-0.43793	-0.42913	9.572e-01	3.369e+00	1.897e+00
3	2.68488	-0.43426	-0.55737	3.126e+00	1.100e+00	5.356e-01
4	3.77169	0.15432	-1.45825	4.310e+00	4.703e+00	1.236e+00
5	3.81436	1.08319	-2.34462	3.412e+00	9.157e+00	9.299e-01
6	3.14263	1.40890	-2.99804	9.248e-03	7.358e+00	7.465e-01
7	3.13808	1.36548	-2.99998	7.435e-05	9.945e-01	4.366e-02
8	3.14146	1.36282	-3.00000	6.576e-07	1.037e+00	4.292e-03
9	3.14145	1.36284	-3.00000	6.000e-07	4.481e-02	1.692e-05
10	3.14145	1.36284	-3.00000	6.000e-07	0.000e+00	0.000e+00

The Final Metric

0.3332	-0.1451
-0.1451	0.6015

Total cpu time (s)= 1.5000

H/W: Try BFGS on these egs

http://www.cse.iitb.ac.in/~CS709/notes/code/unconstrainedOpt/BFGS.m