

HW 4 graded problems 1, 2, 5, 10 pts each

$$1. f(x, y) = \frac{1}{2} [p(x^2 + y^2 - 2x - 2y) + (xy - 1)^2]$$

$$df = [px - p + (xy - 1)y \quad py - p + (xy - 1)x]$$

$$= [-p + px - y + xy^2 \quad -p - x + py + x^2y]$$

$$= (xy - 1)[y \ x] + p[x - 1 \ y - 1]$$

a) $df(x_0, y_0) = 0 \quad \forall p \quad \text{iff} \quad (x_0, y_0) = (1, 1).$

$$d^2f(x_0, y_0) = \begin{bmatrix} p+1 & 1 \\ 1 & p+1 \end{bmatrix} \quad d^2f(x, y) = \begin{bmatrix} p+y^2 & -1+2xy \\ -1+2xy & p+x^2 \end{bmatrix}$$

$$\det(d^2f(x_0, y_0) - \lambda I) = ((p+1) - \lambda)^2 - 1 = \lambda^2 - 2(p+1)\lambda + (p+1)^2$$

~~with eigen~~

eigenvalues of $d^2f(x_0, y_0)$ are $p, p+2$

b) SONC $\iff p \geq 0$

c) SOSC $\iff p > 0$

d) f is convex near (x_0, y_0) if $p > 0$, and

f is not convex near (x_0, y_0) if $p < 0$.

What if $p = 0$?

~~if $p = 0$~~

1. (continued)

If $p=0$, we have

$$f(x, y) = \frac{1}{2}(xy - 1)$$

Let $x = x_0 + dx = 1 + dx$, $y = y_0 + dy = 1 + dy$.

$$f = \frac{1}{2} [dx^2 + 2dx dy + dy^2 + 2dx^2 dy + 2dx dy^2 + dx^2 dy^2]$$

Let $dx = du + dv$, $dy = du - dv$

$$f = \frac{1}{2} [4du^2 + 2du^3 - 2dudv^2 + du^4 - 2du^2 dv^2 + dv^4]$$

Then \forall fixed $du > 0$, f is concave in dv near $dv = 0$. Thus $\forall r > 0$ f is not convex in the ball $B((x_0, y_0), r)$ centered at (x_0, y_0) with radius r .

So f is locally convex at (x_0, y_0) iff $p > 0$.
(another way to do this is to show that, with $p=0$, $\forall \epsilon > 0$
 $\exists (x, y) \in B((x_0, y_0), \epsilon)$ with $\det \nabla^2 f(x, y) < 0$)

e) Newton's method, started sufficiently close to (x_0, y_0) , converges quadratically to (x_0, y_0) iff ~~iff~~

$d^2 f(x_0, y_0)$ is ~~positive definite~~ ~~or~~ ~~negative~~

~~definite~~ ~~iff~~ ~~iff~~ ~~iff~~

nonsingular, i.e., iff $p \neq 0$ and $p \neq -2$

2. Let $A \in \mathbb{R}^{n \times n}$, spd, $b \in \mathbb{R}^n$, $c \in \mathbb{R}$,

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

For $x_0 \in \mathbb{R}^n$, and $p_0, \dots, p_{k-1} \in \mathbb{R}^n$ linearly independent define

$$V_k = \left\{ x \in \mathbb{R}^n : x = x_0 + \sum_{i=0}^{k-1} \alpha_i p_i \text{ for some } \alpha \in \mathbb{R}^k \right\}$$

so that the map

$$\mathbb{R}^k \rightarrow V_k, \alpha \rightarrow P_k \alpha, \quad P_k = \begin{bmatrix} p_0 & \dots & p_{k-1} \end{bmatrix}$$

is a bijection. Define $g_k: \mathbb{R}^k \rightarrow \mathbb{R}$ by

$$g_k(\alpha) = f(x_0 + P_k \alpha)$$

Then $x_0 + P_k \alpha^*$ is the unique minimizer of the restriction of f to V_k if and only if α^* is the unique minimizer of g_k . We now show that the latter condition holds with

$$\alpha^* = (P_k^T A P_k)^{-1} P_k^T (b - A x_0).$$

For this, note that α^* is the only solution

2. (continued)

of the FONC

$$0 = \frac{\partial g_k}{\partial \alpha}(\alpha) = P_k^T A P_k \alpha + P_k^T (A x_0 - b)$$

and that the ~~FOC~~ SOSC

$$0 < \frac{\partial^2 g_k}{\partial \alpha^2}(\alpha) = P_k^T A P_k$$

does hold at $\alpha^* = P_k^T A P_k$ is psd because

A is psd ($v^T P_k^T A P_k v = (P_k v)^T A P_k v \geq 0$). And

$P_k^T A P_k$ is pd because A is pd and P_k is full

rank (if $v^T P_k^T A P_k v = 0$ then $(P_k v)^T A P_k v = 0$,

so $P_k v = 0$, so $v = 0$).

Now suppose

$$x_{k+1} = x_k - \alpha_k P_k, \quad \alpha_k = \frac{(A x_k - b)^T P_k}{(A P_k)^T P_k}$$

$$k = 1, 2, \dots, n$$

and P_0, \dots, P_{n-1} are A -conjugate. Then

$$P_k^T A P_k = I$$

so the unique minimizer of f restricted to V_k is

$$\begin{aligned}x_k^* &= x_0 + P_k (P_k^T A P_k)^{-1} P_k^T (b - A x_0) \\ &= x_0 + P_k P_k^T (b - A x_0)\end{aligned}$$

and furthermore

$$(A P_k)^T P_k = 1$$

so

$$\alpha_k = (A x_k - b)^T P_k.$$

We show by induction that ~~$x_k = x_k^*$~~ $x_k = x_k^*$,

$k = 1, \dots, n$, and hence that ~~$f(x_k) = \min_{x \in V_k} f(x)$~~

$$f(x_k) = \min \{ f(x) : x \in V_k \}.$$

$$\underline{k=1} : x_{1}^* = x_0 + P_1 P_1^T (b - A x_0) = x_0 + p_0 p_0^T (b - A x_0)$$

$$= \cancel{x_0} = x_0 - \left[(A x_0 - b)^T p_0 \right] p_0$$

$$= x_0 - \alpha_0 p_0 = x_1.$$

#

Assume $x_k = x_k^*$, show $x_{k+1} = x_{k+1}^*$.

For this,

$$\begin{aligned}x_{k+1} &= x_k - \alpha_k P_k \\&= x_k^* + \left[(b - Ax_k)^T P_k \right] P_k \\&= x_0 + P_k P_k^T (b - Ax_0) + P_k P_k^T (b - Ax_k) \\&= x_0 + P_k P_k^T (b - Ax_0) + P_k P_k^T (b - Ax_0) \\&\quad + P_k P_k^T A \sum_{i=1}^k (x_i - x_{i-1}) \\&= x_0 + \begin{bmatrix} P_k & P_k \end{bmatrix} \begin{bmatrix} P_k^T \\ P_k^T \end{bmatrix} (b - Ax_0) \\&= x_0 + P_{k+1} P_{k+1}^T (b - Ax_0) \\&= x_{k+1}^*\end{aligned}$$

where we used

$$P_k^T A (x_i - x_{i-1}) = P_k^T A \alpha_{i-1} P_{i-1} = 0, \quad i=1, \dots, k.$$

notes on problem 5

problem 5

The discussion part of this problem is very important to me. You should try to come up with an explanation for the results you report and give evidence or plausible reasoning for your explanation. Believe it or not, there is a lot of explanation possible for this problem.

One person graphed the execution times of various methods on various problem sizes. I really appreciate that. Most people reported best results with TNewton. Some people recommended using LBFGS.

the Hessian H of obj_g is dense - it has roughly n^2 entries. The approximate inverse Hessian B maintained by BFGS is dense also. This explains why Newton takes a long time, as the required Cholesky factorization is $O(n^3)$ operations. But both TNewton and BFGS need to compute products $A*v$ for vectors v and dense matrices A , so why does TNewton outperform BFGS?

If your BFGS code is taking a very long time, you may be performing the update inefficiently, by taking products of dense matrices. (How many arithmetic operations are required to multiply two n -by- n dense matrices? More than n^2 .)

Professor Ferris has indicated a way of performing the update using only a few matrix-vector products with the matrix B . This update, and thus the whole step, requires only $O(n^2)$ operations.

If you are updating B correctly and BFGS is still taking much longer than TNewton, and BFGS takes many more steps to get the same level of accuracy, then it could be that the BFGS approximation is not accurate enough. Ultimately this reflects the inaccuracy of the inverse Hessian approximation, which means BFGS has only superlinear convergence while TNewton can achieve quadratic convergence when the accuracy of the CG solves is increased appropriately as the gradient decreases to 0.

Although H is dense, it has a special structure. It is the sum of a sparse matrix H_1 and a low-rank matrix H_2 . You can use this property to perform efficient evaluations of $H*v$ without ever storing a dense matrix.

This would make TNewton even better. Furthermore, H_1 is block diagonal with 2-by-2 blocks and H_2 is rank 1. Maybe you can use the Sherman-Morrison-Woodbury formula, adding positive diagonal elements to H_1 as necessary to ensure sufficient positive definiteness and thus descent. Is it possible to say anything about the eigenvalues of H ? If your TNewton code is especially fast, how many CG iterations do you need to do each step?

One last comment: Newton's method never computes the inverse of H . Rather, it solves the system $0 = g + H * dx$.