hw 2b) requires $(A \otimes B)(C \otimes D) = AC \otimes BD$

should prove this identity on the hw

back to optimization...

* algorithms for finding minima of

$f(x)$ $\quad f: \mathbb{R}^n \to \mathbb{R}$

structure of most algorithms goes like this:

start w/ some x_0

for $k = 0, 1, 2, \dots$ until convergence

- compute $g_k = \nabla f(\vec{x}_k)$

newton's method

maybe compute the hesian (newton's method)

$H_k = \nabla^2 f(\vec{x}_k)$ & solve

$H_k d_k = -g_k$ for d_k

deriv. of f wrt each component of f. ie, x is a vector

OR

steepest descent

$d_k = -g_k$

$x_{k+1} = x_k + \alpha_k d_k$, for some α_k T.B.D.

how to determine "convergence"?

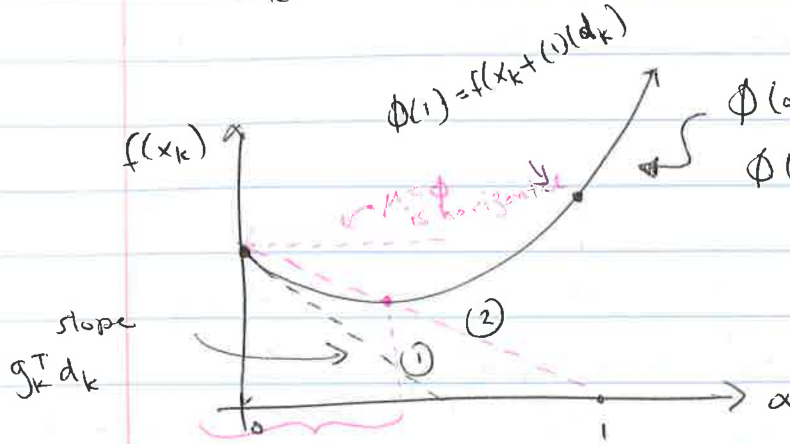
also, note that w/ hesian, (newton's method) need to check if $g_k^T d_k < 0 \Rightarrow$ we indeed have a descent direction. (it's guaranteed / by def$^{\underline{n}}$ for steepest descent).

if we know we're working w/ a descent direction, then for a reasonable choice of alpha we're going to go down.

Convergence test: $\|g_k\|$ is "small enough".
small enough might be abs. or relative error.
ie. $\dfrac{\|g_k\|}{\|f(x_k)\|} \leq \tau$, tolerance, which depends on the problem / context.

Cost - assume we have an analytic formula that includes expressions for ∇f & $\nabla^2 f$.
there are n iterations for steepest descent, n^2 for newton (? i think this is what he said...).

So, how to choose α_k? (assume we have a d_k already - by whatever method).



• this is the region where criteria is satisfied.

$\phi(\alpha) = f(x_k + \alpha d_k)$

$\phi(0) = f(x_k)$

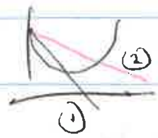
$\boxed{\text{Linesearch Method}}$

in this example $\phi(1) > f(x_k)$, so we want to pick another version of α. $\boxed{\text{Strategy?}}$

Taylor Series:

$$\phi(\alpha) = f(x_k + \alpha d_k) = \underbrace{f(x_k) + \alpha g_k^T d_k}_{\substack{\text{Line through} \\ (0,\ f(x_k))\ \text{w/} \\ \text{slope } g_k^T d_k\ (<0)}} + \underbrace{O(\alpha^2)}_{\text{ignore}}$$

since we know $d_k < 0$, since Line (1) is below function f & tangent to f @ 0, then if we create a new line w/ a larger (but still -ive) slope, then it is guaranteed to be above $\hat{f} \approx f$ for at least some period of time. (even if you had something like



where f goes below line ①).

the pink dashed line is a line through $(0, f(x_k))$ w/ slope $\mu g_k^T d_k$, for $\mu \in (0,1)$. eg. $\mu = 1/2$.

$\mu = 0 \Rightarrow$ horizontal
$\mu = 1 \Rightarrow$ slope is $g_k^T d_k$, same as original line.

Req't on α_k:

$\mu = 1/2$ convention. people research this. (so, fun!)

eq ①

$$\underbrace{f(x_k + \alpha_k d_k)}_{\text{value of } \hat{f} \text{ line at } \alpha_k} \leq \underbrace{f(x_k) + \mu \alpha_k g_k^T d_k}_{\text{value of pink line at } \alpha_k}$$

"simple-minded approach"
- try $\alpha_k = 1$. if req't is satisfied, then accept.
- otherwise, try $\alpha_k \leftarrow \alpha_k / 2$ (reduce α_k)
- iterate...

✱ note: ∃ still more book keeping to do, to ensure we have a global min.

for n-dimensions, think of the 2D graph as a cross-section in the direction of $\vec{d}_k$ (vector).

Typical convergence th'm

For a descent method applied to a function f that is bounded below, for which gradient g is Lipschitz continuous, then $\|g_k\| \to 0$

Lipschitz continuous

$$\equiv \|g(x) - g(y)\| \leq L\|x - y\|$$

for some $L > 0$

$\forall\, x, y$ in the region of interest.

∃ better ways to choose α? viz. simple minded approach discussed earlier - want a diff way that could be smarter ... ie, if we can use our knowledge of f to reduce stupid guesses, we are happier.

in particular, we'll have to compute eqn (1) less often.

Cost of computing ① ? Well,

$f(x_k)$ - free

$\mu \alpha_k g_k^T d_k$ - cheap

but $f(x_k + \alpha_k d_k)$ (LHS) might be arbitrarily complex f^n f.

So, an alternative approach to meet the req't of eq^n ① :

start w/ $\alpha_{k_1} = 1$

$\alpha_{k_2} =$ something like $1/2 \alpha_{k_1}$

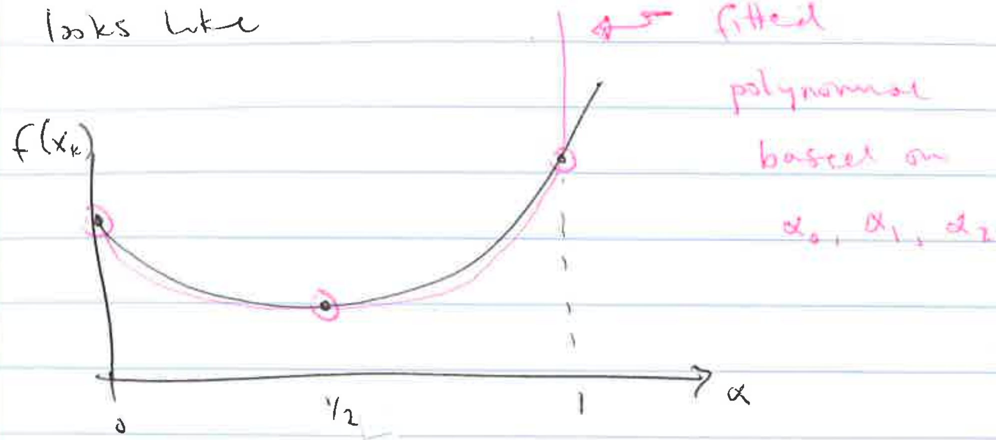
Now, we have candidate values:

$\alpha_k = 0$ $f(x_k)$ $\alpha_k = 1$ $f(x_k + d_k)$

$\alpha = 1/2$ $f(x_k + \frac{1}{2} d_k)$

let $p(\alpha) =$ quadratic polynomial that interpolates 3 points. Then, find α that minimizes $p(\alpha)$. call this α_{k_3}.

looks like



the fitted polynomial makes use of values we've already computed.