Recap: $\underline{x}_{k+1} = \underline{x}_k + p_k^N$

Newton: $p_k^N = -\underbrace{B(\underline{x}_k)^{-1}}_{B_{k+1}\ (\text{NOTATION})} \nabla f(\underline{x}_k)$

BFGS gives an approximation for B_{k+1} based on the secant condition:

$$\underbrace{y_k} = B_{k+1}\, \underline{s}_k \qquad (1)$$

$$y_k = \nabla f(\underline{x}_{k+1}) - \nabla f(\underline{x}_k)$$

$$\underline{s}_k = \underline{x}_{k+1} - \underline{x}_k$$

$$B_{k+1} \simeq B_k + \alpha\, \underbrace{y_k y_k^T + \beta\, (B_k \underline{s}_k)}_{\text{OUTER PRODUCTS}}(B_k \underline{s}_k)^T$$

Last week: Secant condition (1) fixes α and β:

$$\left.\begin{aligned} \beta &= -\frac{1}{\langle \underline{s}_k, B_k^T \underline{s}_k \rangle} \\ \alpha &= \frac{1}{\langle y_k, \underline{s}_k \rangle} \end{aligned}\right\} \quad (2)$$

Theorem 4.1 (Examinable)

Also, the approximate Hessian B_{k+1} has the positive-definite property (Tuesday):

TYPO

Theorem 4.2 Provided B_0 is symmetric and positive definite and the curvature condition $\langle y_k, s_k \rangle > 0$ is satisfied, then the BFGS method produces a symmetric positive-definite approximation to the Hessian at each iteration.

EXAMINABLE

Today: Look at B_{k+1}^{-1}. Typically, inverting an $n \times n$ matrix is an $O(n^3)$ operation, unless the matrix has a special structure. To exploit the special structure of B_{k+1}, we use the Sherman — Morrison — Woodbury formula.

Context: "The Freshman's Dream"

$$a = b + c$$

$$\left.\begin{aligned} a^2 &= b^2 + c^2 \\ \frac{1}{a} &= \frac{1}{b} + \frac{1}{c} \end{aligned}\right\}$$

"Law" of universal is the "Freshman's Dream"

$$\frac{\text{si}\cancel{n}\ x}{\cancel{n}} = \text{six} = 6$$

In the Sherman — Morrison — Woodbury formula, the Freshman's Dream almost comes true!

$$M = B + UV$$

where:

- $M \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times n}$ invertible
- $U \in \mathbb{R}^{n \times k}$, $V \in \mathbb{R}^{k \times n}$

Sherman — Morrison — Woodbury says:

Sherman – Morrison – Woodbury says:

TYPED NOTES ARE CORRECT

$$M^{-1} = B^{-1} - B^{-1} U (I_k + V B^{-1} U)^{-1} V B^{-1}$$

We can apply this to BFGS:

$$B_{k+1} = B_k + \alpha \underline{y}_k \underline{y}_k^T + \beta (B_k \underline{s}_k)(B_k \underline{s}_k)^T$$

Result:

OK ✓

$$\left.\begin{aligned} B_{k+1}^{-1} &= B_k^{-1} + \left(1 + \frac{\langle \underline{y}_k, B_k^{-1} \underline{y}_k \rangle}{\langle \underline{s}_k, \underline{y}_k \rangle}\right) \frac{\underline{s}_k \underline{s}_k^T}{\langle \underline{y}_k, \underline{s}_k \rangle} \\ &\quad - \frac{B_k^{-1} y_k \underline{s}_k^T + \underline{s}_k \underline{y}_k^T B_k^{-1}}{\langle \underline{y}_k, \underline{s}_k \rangle} \end{aligned}\right\} \quad (3)$$

RHS involves only inner and outer products – $O(n^2)$ operations. Hence, computing B_{k+1}^{-1} is reduced from $O(n^3)$ to $O(n^2)$ operation.

One other quasi-Newton method — Barzilai Borwein (§4.3)

$$\underline{s}_k = \underline{x}_k - \underline{x}_{k-1}$$

$$\underline{y}_k = \nabla f(\underline{x}_k) - \nabla f(\underline{x}_{k-1})$$

Secant condition:

$$\underline{y}_k = B_k \underline{s}_k$$

We solve the secant condition in an approximate sense, in a 1D subspace of $\mathbb{R}^{n \times n}$.

Compare:

- BFGS, 3D subspace spanned by B_k, $\underline{y}_k\underline{y}_k^T$ and $(B_k \underline{s}_k)(B_k \underline{s}_k)^T$
- Barzilai-Borwein, 1D subspace spanned by $\mathbb{I}_n$.

Hence,

$$B_k \simeq \lambda \mathbb{I}_n$$

We take:

$$B_k \simeq \frac{1}{\alpha_k} \mathbb{I}_n$$

We fix α_k by demanding that B_k satisfy the secant condition in the least-squares sense:

$$\alpha_k = \arg \min_{\alpha > 0} \| B_k \underline{s}_k - \underline{y}_k \|_2^2 .$$

i.e.

$$\alpha_k = \arg \min_{\alpha > 0} \| \frac{1}{\alpha} \underline{\mathbb{I}_n} \underline{s}_k - \underline{y}_k \|_2^2 . \quad (4)$$

<u>Next</u>: We solve for α_k (p. 34, Examinable)

Let $\beta = 1/\alpha$ in (4). Introduce:

$$\begin{aligned}
\phi(\beta) &= \| \beta \underline{s}_k - \underline{y}_k \|_2^2 \\
&= \langle \beta \underline{s}_k - \underline{y}_k , \beta \underline{s}_k - \underline{y}_k \rangle \\
&= \beta^2 \langle \underline{s}_k, \underline{s}_k \rangle - 2\beta \langle \underline{s}_k, \underline{y}_k \rangle + \langle \underline{y}_k, \underline{y}_k \rangle
\end{aligned}$$

We aim to minimize $\phi(\beta)$. So we compute $\phi'(\beta)$:

$$\phi'(\beta) = 2\beta \langle \underline{s}_k, \underline{s}_k \rangle - 2\langle \underline{s}_k, \underline{y}_k \rangle + 0 .$$

$$\phi'(\beta) = 0 \Rightarrow \cancel{2}\beta \langle \underline{s}_k, \underline{s}_k \rangle - \cancel{2}\langle \underline{s}_k, \underline{y}_k \rangle = 0 .$$

$$\Rightarrow \beta = \frac{\langle \underline{s}_k, \underline{y}_k \rangle}{\langle \underline{s}_k, \underline{s}_k \rangle} \longleftarrow \|\underline{s}_k\|_2^2$$

But $\alpha = 1/\beta$.

So the α that minimizes ϕ is:

$$\alpha = \frac{\|\underline{s}_k\|_2^2}{\langle \underline{s}_k, \underline{y}_k \rangle}$$

Back to

$$\underline{x}_{k+1} = \underline{x}_k - B_k^{-1} \nabla f(\underline{x}_k)$$

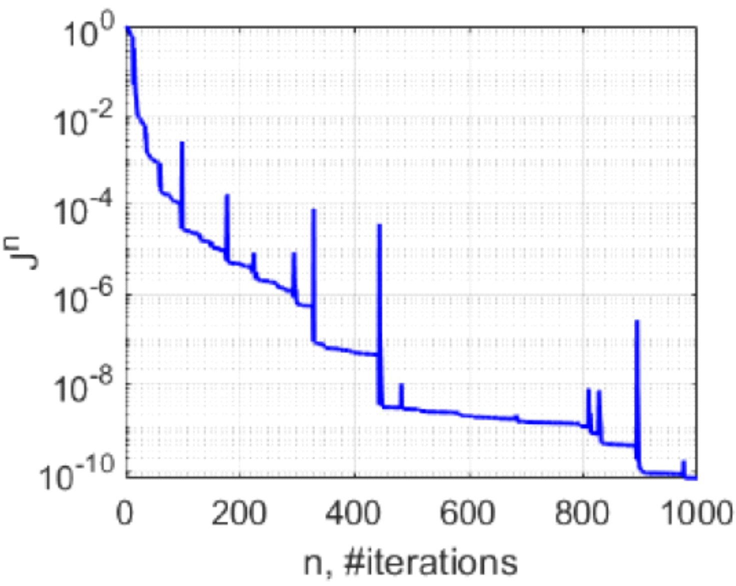
$$\simeq \underline{x}_k - \left(\frac{1}{\alpha_k} \mathbb{I}_n\right)^{-1} \underline{\nabla} f(\underline{x}_k)$$

$$= \underline{x}_k - \alpha_k \nabla f(\underline{x}_k)$$

So Barzilai – Borwein reduces to steepest descent, with a procedure for computing the step length α_k.

<u>Warning</u>: BB does not reduce the cost function at each iteration, so it is not guaranteed that

$$f(\underline{x}_{k+1}) < f(\underline{x}_k).$$



← Front cover of lecture notes, Barzilai–Borwein algorithm in action.

Summary so far:

- Different methods for computing the search direction p_k:
  - SD
  - Newton
  - Quasi-Newton
    - + BFGS
    - + BB
- Mentioned last week how — in general — the steplength is computed from:

$$\alpha_k = \arg\min_{\alpha > 0} f(\underline{x}_k + \alpha p_k) \qquad (5)$$

In the next part of the course (Chapter 5) we look at methods for solving (5). Plan for look at methods for solving (S). Plan for next few lectures:

- Methods for solving (S) approximately
- Convergence proofs next Tuesday.
- Wrap up the convergence proof next Thursday.
- Look at Exercises # 1 next Thursday (Laptops)

The Strong Wolfe Conditions (SWCs) give a satisfactory approximate solution to (S).

Rationale for the SWCs:

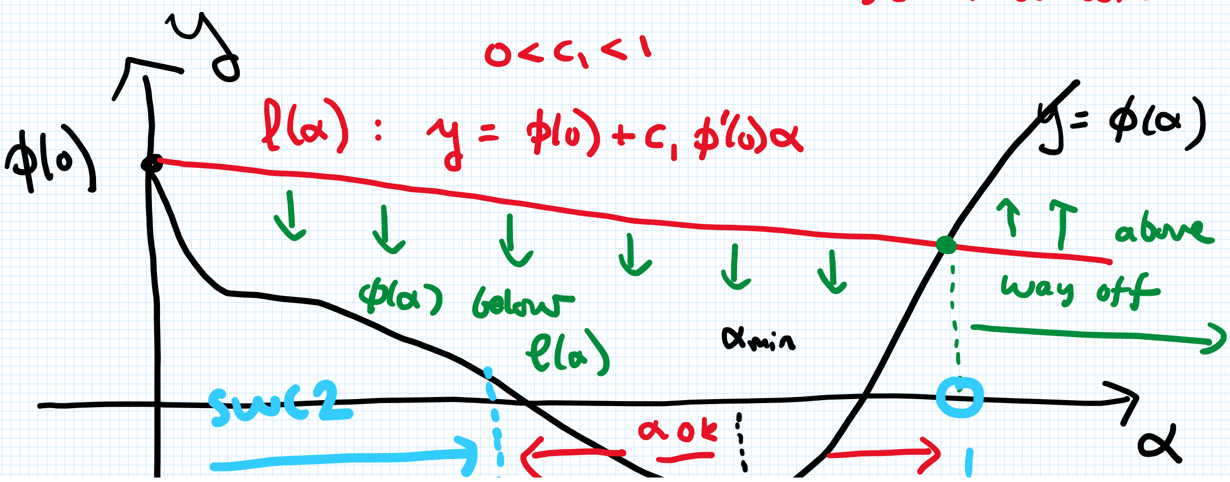
$$\phi(\alpha) = f(\underbrace{\underline{x}_k}_{\text{previous iteration}} + \alpha \underbrace{p_k}_{\text{search direction}})$$

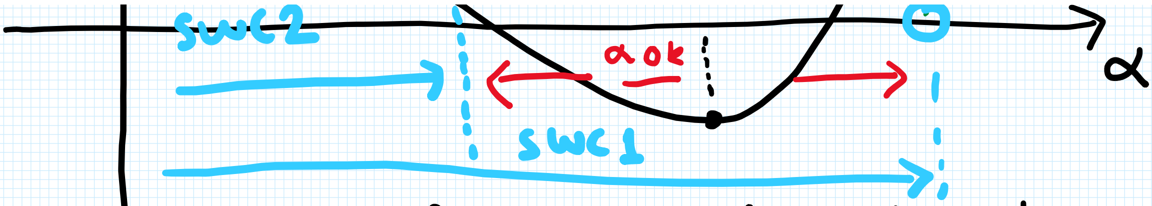
The aim of the SWCs is to find an α that gets close to the minimum of ϕ.

Check $\phi'(\alpha) = p_k \cdot \nabla f(\underline{x}_k + \alpha p_k)$

$$\boxed{\phi'(0) = p_k \cdot \nabla f(\underline{x}_k)} < 0$$

by choice of search direction





Aim of SWCs is to find a suitable α, close to α_{min}.

keep $\phi(\alpha)$ below $\ell(\alpha)$:

$$\phi(\alpha) < \ell(\alpha)$$

$$\Rightarrow \boxed{\phi(\alpha) < \phi(0) + c_1 \phi'(0)\,\alpha} \qquad \text{SWC 1}$$

~~S~~ SWC 2: $\phi'(\alpha_{min}) = 0$ (@ min)

SWC 2: $|\phi'(\alpha)|$ should be small.

$$\boxed{|\phi'(\alpha)| < c_2 |\phi'(0)|, \quad c_2 \in (0,1).}$$

We require: $0 < c_1 < c_2 < 1$.

<u>Lecture 3</u>: Does an α satisfying the SWCs always exist? Yes. See Theorem 5.1.

Theorem 5.1 Let $\phi(\alpha)$ be a continuously differentiable function which is bounded below: $\phi(\alpha) \geq \phi_{min}$. If $0 < c_1 < c_2 < 1$, then there is always an α satisfying the SWCs.

Assume $\phi'(0) < 0$.

Proof in two parts: Take $0<c_1<1$. Introduce

$$\Delta(\alpha) = \ell(\alpha) - \phi(\alpha)$$

$$= [\phi(0) + c_1\alpha\phi'(0)] - \phi(\alpha)$$

$$\phi(\alpha) \geq \phi_{min} \Rightarrow -\phi(\alpha) \leq -\phi_{min}$$

Hence:

$$\Delta(\alpha) \leq \underbrace{[\phi(0) + c_1\alpha\underbrace{\phi'(0)}_{\text{neg}}] - \phi_{min}}_{\to -\infty \text{ as } \alpha \to \infty}.$$

Hence, $\boxed{\Delta(\alpha) \to -\infty \text{ as } \alpha \to \infty.}$ (i)

Also, look at

$$\frac{d\Delta}{d\alpha} = \frac{d\ell}{d\alpha} - \phi'(\alpha).$$

$$= c_1\phi'(0) - \phi'(\alpha)$$

$$\left.\frac{d\Delta}{d\alpha}\right|_{\alpha=0} = c_1\phi'(0) - \phi'(0).$$

$$= \underbrace{(c_1-1)}_{\text{NEG}}\ \underbrace{\phi'(0)}_{\text{NEG}} > 0$$

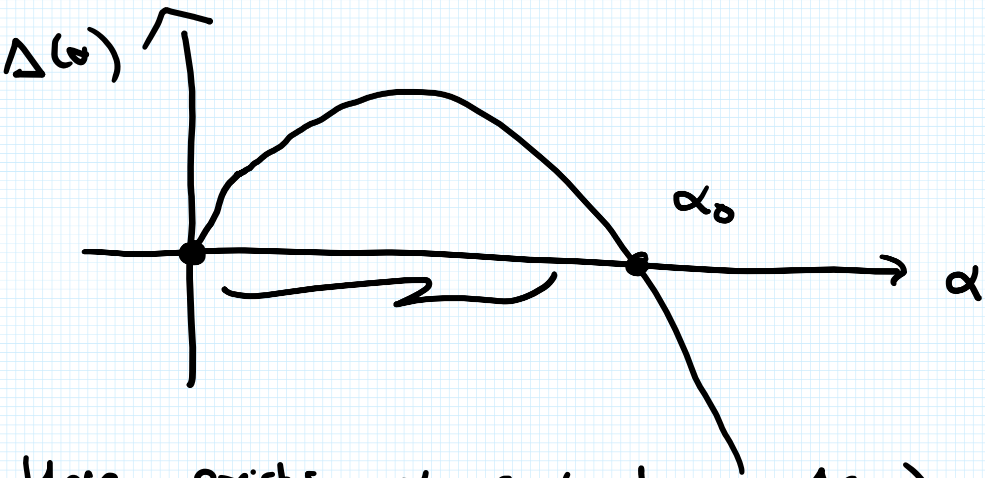
$$\boxed{\left.\frac{d\Delta}{d\alpha}\right|_{\alpha=0} > 0}$$ (ii)

$$\Delta(\alpha=0) = \ell(0) - \phi(0)$$

$$\Delta(\alpha=0) = \ell(0) - \phi(0)$$
$$= [\phi(0) + c_1 \phi'(0) \cdot 0] - \phi(0)$$

$\therefore$ $\boxed{\Delta(\alpha=0) = 0}$ (iii)

Combine (i), (ii) and (iii) into a plot:



So there exists α_0 such that $\Delta(\alpha_0) = 0$.

Furthermore, $\Delta(\alpha) > 0 \ \forall \ \alpha \in (0, \alpha_0)$.

Hence, $\ell(\alpha) - \phi(\alpha) > 0 \ \forall \ \alpha \in (0, \alpha_0)$

$$\Rightarrow \boxed{\phi(\alpha) < \phi(0) + c_1 \phi'(0)\alpha \quad \forall \ \alpha \in (0, \alpha_0)}$$

Hence, for $0 < c_1 < 1$, SWC 1 holds for all α between 0 and α_0.

Second part of Theorem. We have:

$$\Delta(\alpha_0) = 0.$$
$$\Rightarrow \ell(\alpha_0) - \phi(\alpha_0) = 0$$

$$\Rightarrow \quad \phi(\alpha_0) = \phi(0) + c_1 \alpha_0 \phi'(0) \quad . (iv)$$

Apply Taylor's Remainder Theorem: There exists $\beta \in (0, \alpha_0)$ such ~~that~~:

$$\phi(\alpha_0) = \phi(0) + \alpha_0 \phi'(\beta) \quad . (v)$$

Combine (iv) and (v)

$$\phi'(\beta) = c_1 \phi'(0)$$

Take absolute values: $c_2 > c_1$

$$|\phi'(\beta)| = c_1 |\phi'(0)| < c_2 |\phi'(0)|$$

Hence, SWC 2 holds for β:

$$|\phi'(\beta)| < c_2 |\phi'(0)|$$

Because SWC 1 holds for all $\alpha \in (0, \alpha_0]$, and $\beta_0 \in (0, \alpha_0)$, SWC1 holds for β_0:

$$\phi(\beta) \leq \phi(0) + c_1 \phi'(0) \cdot \beta$$

Therefore, β exists (by Taylor's Remainder Theorem) and satisfies SWC 1 and SWC 2.

<u>Remark</u>: It is good to work with the SWCs because SWC2 does not depend on the sign of derivatives. Some less restrictive conditions for

derivatives. Some less restrictive conditions for an α-value approximating α_{min} are the Wolfe conditions:

$$\phi(\alpha) \leq \phi(0) + c_1 \phi'(0)\, \alpha$$

$$\phi'(\alpha) > c_2\, \phi'(0)$$

$$0 < c_1 < c_2 < 1$$

Another method for finding the stepsize is very easy to code but does not have a theoretical underpinning — Backtracking Line search (§5.3)

Pseudocode:

Choose an initial guess for α_k, call it α.

Fix $\rho \in (0,1)$, $c \in (0,1)$.

Outer loop over k.

Inner "while" loop:

while $f(\underline{x}_k + \alpha p_k) > f(\underline{x}_k) + c\, p_k \cdot \nabla f(\underline{x}_k)$

$\alpha \rightarrow \alpha \cdot \rho$

end while