

Steepest descent

Armijo backtracking

Steepest descent Armijo backtracking

Line-Search Methods for Smooth Unconstrained Optimization

Daniel P. Robinson
Department of Applied Mathematics and Statistics
Johns Hopkins University

September 17, 2020

1 / 106

The problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) \quad \rightarrow \quad \nabla f(x) = 0$$

where the objective function $f : \mathbb{R}^n \rightarrow \mathbb{R}$

- assume that $f \in C^1$ (sometimes C^2) and is Lipschitz continuous
- in practice this assumption may be violated, but the algorithms we develop may still work
- in practice it is very rare to be able to provide an explicit minimizer
- we consider iterative methods: given starting guess x_0 , generate sequence $\{x_k\}$ for $k = 1, 2, \dots$
- AIM:** ensure that (a subsequence) has some favorable limiting properties
 - satisfies first-order necessary conditions
 - satisfies second-order necessary conditions

Notation: $f_k = f(x_k)$, $g_k = g(x_k)$, $H_k = H(x_k)$

Outline

- 1 Generic Linesearch Framework
- 2 Computing a descent direction p_k (search direction)
 - Steepest descent direction
 - Modified Newton direction
 - Quasi-Newton directions for medium scale problems
 - Limited-memory quasi-Newton directions for large-scale problems
 - Linear CG method for large-scale problems
- 3 Choosing the step length α_k (linesearch)
 - Backtracking-Armijo linesearch
 - Wolfe conditions
 - Strong Wolfe conditions
- 4 Complete Algorithms
 - Steepest descent backtracking Armijo linesearch method
 - Modified Newton backtracking-Armijo linesearch method
 - Modified Newton linesearch method based on the Wolfe conditions
 - Quasi-Newton linesearch method based on the Wolfe conditions

2 / 106

The basic idea

- consider descent methods, i.e., $f(x_{k+1}) < f(x_k)$
- calculate a search direction p_k from x_k
- ensure that this direction is a descent direction, i.e.,

$$g_k^T p_k < 0 \quad \text{if } g_k \neq 0$$

so that, for small steps along p_k , the objective function f will be reduced

- calculate a suitable steplength $\alpha_k > 0$ so that

$$f(x_k + \alpha_k p_k) < f_k$$

- computation of α_k is the linesearch
- the computation of α_k may itself require an iterative procedure
- generic update for linesearch methods is given by

$$x_{k+1} = x_k + \alpha_k p_k$$

5 / 106

Issue 1: steps might be too long

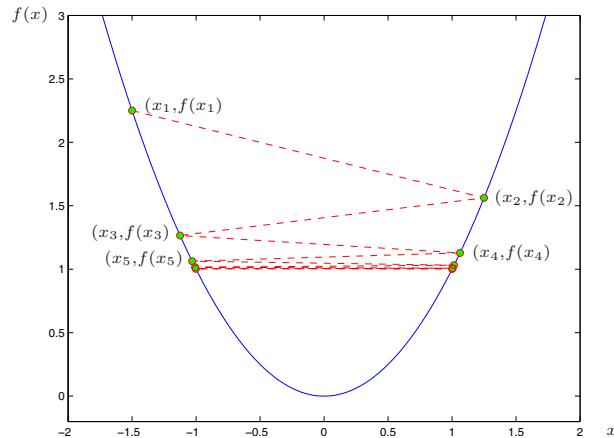


Figure: The objective function $f(x) = x^2$ and the iterates $x_{k+1} = x_k + \alpha_k p_k$ generated by the descent directions $p_k = (-1)^{k+1}$ and steps $\alpha_k = 2 + 3/2^{k+1}$ from $x_0 = 2$.

- decrease in f is **not** proportional to the size of the directional derivative!

6 / 106

What is a practical linesearch?

- in the “early” days **exact** linesearches were used, i.e., pick α_k to minimize

$$f(x_k + \alpha p_k)$$

- an exact linesearch requires univariate minimization
- cheap if f is simple, e.g., a quadratic function
- generally **very expensive** and not cost effective
- exact linesearch may not be much better than an approximate linesearch

- modern methods use **inexact** linesearches

- ensure steps are neither too long nor too short
- make sure that the decrease in f is proportional to the directional derivative
- try to pick “appropriate” initial stepsize for fast convergence
 - related to how the search direction s_k is computed

- the descent direction (search direction) is also important

- “bad” directions may not converge at all
- more typically, “bad” directions may converge **very** slowly

Issue 2: steps might be too short

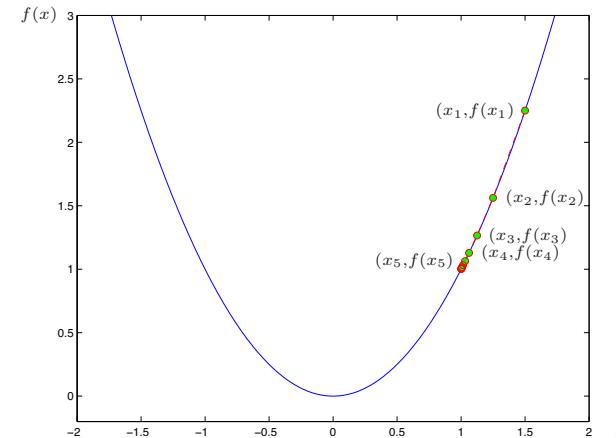


Figure: The objective function $f(x) = x^2$ and the iterates $x_{k+1} = x_k + \alpha_k p_k$ generated by the descent directions $p_k = -1$ and steps $\alpha_k = 1/2^{k+1}$ from $x_0 = 2$.

- decrease in f is **not** proportional to the size of the directional derivative!

7 / 106

gradient is the direction where function increase at fast rate

Definition 2.1 (Steepest descent direction)

For a differentiable function f , the search direction

$$p_k \stackrel{\text{def}}{=} -\nabla f(x_k) \equiv -g_k \rightarrow \text{decrease at fastest.}$$

is called the **steepest-descent** direction.

- p_k is a descent direction provided $g_k \neq 0$
 - $g_k^T p_k = -g_k^T g_k = -\|g_k\|_2^2 < 0$
- p_k solves the problem

$$\underset{p \in \mathbb{R}^n}{\text{minimize}} \quad m_k^L(x_k + p) \quad \text{subject to} \quad \|p\|_2 = \|g_k\|_2$$

- $m_k^L(x_k + p) \stackrel{\text{def}}{=} f_k + g_k^T p$
- $m_k^L(x_k + p) \approx f(x_k + p)$

Any method that uses the steepest-descent direction is a **method of steepest descent**.

Observation: the steepest descent direction is also the **unique** solution to

$$\underset{p \in \mathbb{R}^n}{\text{minimize}} \quad f_k + g_k^T p + \frac{1}{2} p^T p \quad \xrightarrow{\text{quadratic function in } p} \text{solution is } -g_k$$

- approximates second-derivative Hessian by the identity matrix \mathbf{I}

- how often is this a good idea?

- is it a surprise that convergence is typically **very slow**?

Idea: choose **positive definite** B_k and compute search direction as the **unique** minimizer

better approximation $p_k = \underset{p \in \mathbb{R}^n}{\text{argmin}} \quad m_k^0(p) \stackrel{\text{def}}{=} f_k + g_k^T p + \frac{1}{2} p^T B_k p$ quadratic convex function

- p_k satisfies $B_k p_k = -g_k$

- why must B_k be positive definite?

- ▶ $B_k \succ 0 \implies M_k^0$ is strictly convex \implies unique solution
- ▶ if $g_k \neq 0$, then $p_k \neq 0$ and is a descent direction

positive definite

$$p_k^T g_k = -p_k^T B_k p_k < 0 \implies p_k \text{ is a descent direction}$$

- pick "intelligent" B_k that has "useful" curvature information

- **IF** $H_k \succ 0$ and we choose $B_k = H_k$, then p_k is the Newton direction. Awesome!

use different P_k might give you faster convergence

Method 1: small scale problems ($n < 1000$)

Algorithm 1 Compute modified Newton matrix B from H

```

1: input  $H$ 
2: Choose  $\beta > 1$ , the desired bound on the condition number of  $B$ .
3: Compute the spectral decomposition  $H = V\Lambda V^T$ .
4: if  $H = 0$  then
5:   Set  $\varepsilon = 1$ 
6: else
7:   Set  $\varepsilon = \|H\|_2/\beta > 0$ 
8: end if
9: Compute
```

$$\bar{\Lambda} = \text{diag}(\bar{\lambda}_1, \bar{\lambda}_2, \dots, \bar{\lambda}_n) \quad \text{with} \quad \bar{\lambda}_j = \begin{cases} \lambda_j & \text{if } \lambda_j \geq \varepsilon \\ \varepsilon & \text{otherwise} \end{cases}$$

10: **return** $B = V\bar{\Lambda}V^T \succ 0$, which satisfies $\text{cond}(B) \leq \beta$

- replaces the eigenvalues of H that are "not positive enough" with ε
- $\bar{\Lambda} = \Lambda + D$, where

$$D = \text{diag}(\max(0, \varepsilon - \lambda_1), \max(0, \varepsilon - \lambda_2), \dots, \max(0, \varepsilon - \lambda_n)) \succeq 0$$

- $B = H + E$, where $E = VDV^T \succeq 0$

Question: How do we choose the positive-definite matrix B_k ?

Ideally, B_k is chosen such that

- $\|B_k - H_k\|$ is "small"
- $B_k = H_k$ when H_k is "sufficiently" positive definite.

Comments:

- for the remainder of this section, we omit the suffix k and write $H = H_k$, $B = B_k$, and $g = g_k$.
- for a symmetric matrix $A \in \mathbb{R}^{n \times n}$, we use the matrix norm $\|A\|_2 = \max_{1 \leq j \leq n} |\lambda_j|$

with $\{\lambda_j\}$ the eigenvalues of A .

- the **spectral decomposition** of H is given by $H = V\Lambda V^T$, where H is symmetric

Expensive

$$V = (v_1 \ v_2 \ \dots \ v_n) \quad \text{and} \quad \Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$$

with $Hv_j = \lambda_j v_j$ and $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$.

- H is positive definite if and only if $\lambda_j > 0$ for all j .
- computing the spectral decomposition is, generally, **very expensive**!

14/106

$$B_k p_k = -g_k \iff p_k = -B_k^{-1} g_k$$

Question: What are the properties of the resulting search direction?

$$p = -B^{-1}g = -V\bar{\Lambda}^{-1}V^T g = -\sum_{j=1}^n \frac{v_j^T g}{\bar{\lambda}_j} v_j$$

Taking norms and using the orthogonality of V , gives

$$\begin{aligned} \|p\|_2^2 &= \|B^{-1}g\|_2^2 &= \sum_{j=1}^n \frac{(v_j^T g)^2}{\bar{\lambda}_j^2} \\ &= \sum_{\lambda_j \geq \varepsilon} \frac{(v_j^T g)^2}{\lambda_j^2} + \sum_{\lambda_j < \varepsilon} \frac{(v_j^T g)^2}{\varepsilon^2} \end{aligned}$$

Issue

p might have large weight at those direction with $\lambda \leftarrow$ or close to 0 because $\frac{1}{\lambda^2}$ is large

Thus, we conclude that

$$\|p\|_2 = O\left(\frac{1}{\varepsilon}\right) \quad \text{provided } v_j^T g \neq 0 \text{ for at least one } \lambda_j < \varepsilon$$

- the step p goes unbounded as $\varepsilon \rightarrow 0$ provided $v_j^T g \neq 0$ for at least one $\lambda_j < \varepsilon$
- any **indefinite** matrix will generally satisfy this property
- the next method that we discuss is better!

Method 2: small scale problems ($n < 1000$)

Better one

Algorithm 2 Compute modified Newton matrix B from H

- 1: **input** H
- 2: Choose $\beta > 1$, the desired bound on the condition number of B .
- 3: Compute the spectral decomposition $H = V\Lambda V^T$.

4: **if** $H = \mathbf{0}$ **then**

5: Set $\varepsilon = 1$

6: **else**

7: Set $\varepsilon = \|H\|_2/\beta > 0$

8: **end if**

9: Compute

$$\bar{\Lambda} = \text{diag}(\bar{\lambda}_1, \bar{\lambda}_2, \dots, \bar{\lambda}_n) \quad \text{with} \quad \bar{\lambda}_j = \begin{cases} \lambda_j & \text{if } \lambda_j \geq \varepsilon \\ -\lambda_j & \text{if } \lambda_j \leq -\varepsilon \\ \varepsilon & \text{otherwise} \end{cases}$$

10: **return** $B = V\bar{\Lambda}V^T \succ 0$, which satisfies $\text{cond}(B) \leq \beta$

- replace small eigenvalues λ_i of H with ε
 - replace “sufficiently negative” eigenvalues λ_i of H with $-\lambda_i$
 - $\bar{\Lambda} = \Lambda + D$, where
- $$D = \text{diag}(\max(0, -2\lambda_1, \varepsilon - \lambda_1), \dots, \max(0, -2\lambda_n, \varepsilon - \lambda_n)) \succeq 0$$
- $B = H + E$, where $E = VDV^T \succeq 0$

17 / 106

Suppose that $B = H + E$ is computed from Algorithm 2 so that

$$E = B - H = V(\bar{\Lambda} - \Lambda)V^T$$

and since V is orthogonal that

$$\|E\|_2 = \|V(\bar{\Lambda} - \Lambda)V^T\|_2 = \|\bar{\Lambda} - \Lambda\|_2 = \max_j |\bar{\lambda}_j - \lambda_j|$$

By definition

$$\bar{\lambda}_j - \lambda_j = \begin{cases} 0 & \text{if } \lambda_j \geq \varepsilon \\ \varepsilon - \lambda_j & \text{if } -\varepsilon < \lambda_j < \varepsilon \\ -2\lambda_j & \text{if } \lambda_j \leq -\varepsilon \end{cases}$$

which implies that

$$\|E\|_2 = \max_{1 \leq j \leq n} \{0, \varepsilon - \lambda_j, -2\lambda_j\}.$$

However, since $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$, we know that

$$\|E\|_2 = \max \{0, \varepsilon - \lambda_n, -2\lambda_n\}$$

- if $\lambda_n \geq \varepsilon$, i.e., H is sufficiently positive definite, then $E = \mathbf{0}$ and $B = H$
- if $\lambda_n < \varepsilon$, then $B \neq H$ and it can be shown that $\|E\|_2 \leq 2 \max(\varepsilon, |\lambda_n|)$
- regardless, B is well-conditioned by construction

18 / 106

Example 2.2

Consider

$$g = \begin{pmatrix} 2 \\ 4 \end{pmatrix} \quad \text{and} \quad H = \begin{pmatrix} 1 & 0 \\ 0 & -2 \end{pmatrix} \quad \text{indefinite matrix}$$

- The Newton direction is $p^N = -H^{-1}g = \begin{pmatrix} -2 \\ 2 \end{pmatrix}$ so that $g^T p^N = 4 > 0$ (ascent direction)

and p^N is a saddle point of the quadratic model $g^T p + \frac{1}{2}p^T H p$ since H is indefinite.

- Algorithm 1 (Method 1) generates

$$B = \begin{pmatrix} 1 & 0 \\ 0 & \varepsilon \end{pmatrix}$$

so that

$$p = -B^{-1}g = \begin{pmatrix} -2 \\ -\frac{4}{\varepsilon} \end{pmatrix} \quad \text{and} \quad g^T p = -4 - \frac{16}{\varepsilon} < 0 \quad (\text{descent direction})$$

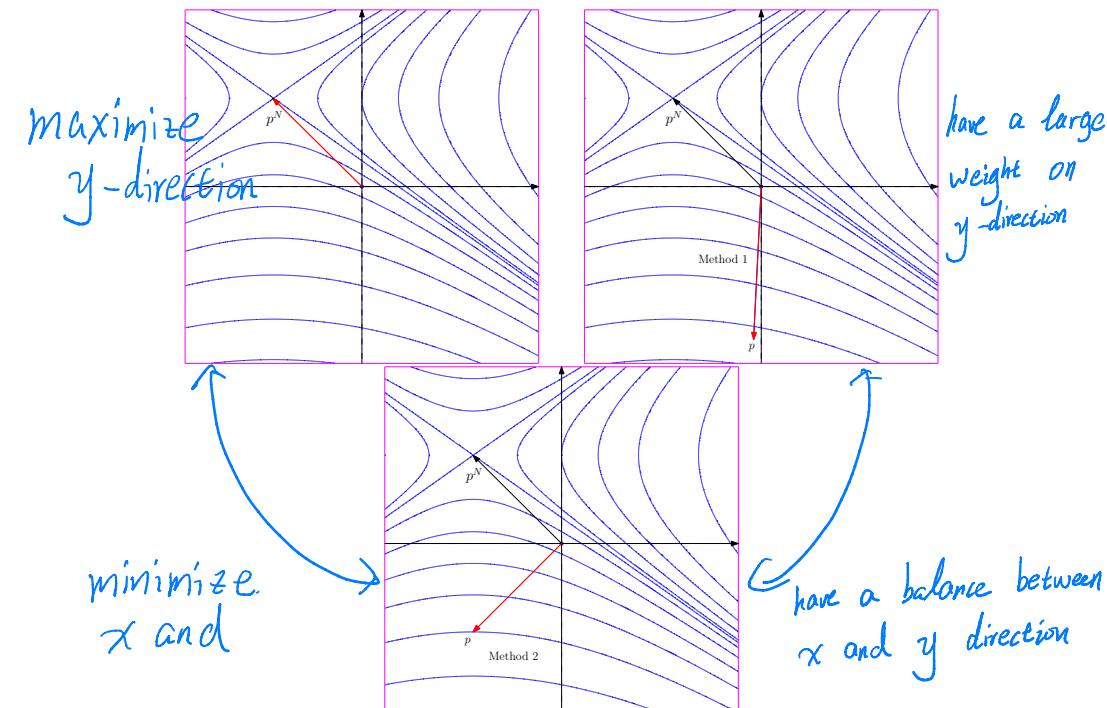
- Algorithm 2 (Method 2) generates

$$B = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$$

so that

$$p = -B^{-1}g = \begin{pmatrix} -2 \\ -2 \end{pmatrix} \quad \text{and} \quad g^T p = -12 < 0 \quad (\text{descent direction})$$

19 / 106



Question: What is the geometric interpretation?

Answer: Let the spectral decomposition of \mathbf{H} be

$$\mathbf{H} = \mathbf{V}\Lambda\mathbf{V}^T$$

and assume that $\lambda_j \neq 0$ for all j . Partition the eigenvector and eigenvalue matrices as

$$\Lambda = \begin{pmatrix} \Lambda_+ & \\ & \Lambda_- \end{pmatrix} \quad \begin{array}{c} \} \lambda_j > 0 \\ \} \lambda_j < 0 \end{array} \quad \text{and} \quad \mathbf{V} = \begin{pmatrix} \mathbf{V}_+ & \mathbf{V}_- \end{pmatrix} \quad \begin{array}{c} \} \lambda_j > 0 \\ \} \lambda_j < 0 \end{array}$$

So that $\mathbf{H} = (\mathbf{V}_+ \quad \mathbf{V}_-) \begin{pmatrix} \Lambda_+ & \\ & \Lambda_- \end{pmatrix} \begin{pmatrix} \mathbf{V}_+^T \\ \mathbf{V}_-^T \end{pmatrix}$ and the Newton direction is

$$\begin{aligned} \mathbf{p}^N &= -\mathbf{V}\Lambda^{-1}\mathbf{V}^T\mathbf{g} = -(\mathbf{V}_+ \quad \mathbf{V}_-) \begin{pmatrix} \Lambda_+^{-1} & \\ & \Lambda_-^{-1} \end{pmatrix} \begin{pmatrix} \mathbf{V}_+^T \\ \mathbf{V}_-^T \end{pmatrix} \mathbf{g} \\ &= -(\mathbf{V}_+ \quad \mathbf{V}_-) \begin{pmatrix} \Lambda_+^{-1} & \\ & \Lambda_-^{-1} \end{pmatrix} \begin{pmatrix} \mathbf{V}_+^T\mathbf{g} \\ \mathbf{V}_-^T\mathbf{g} \end{pmatrix} \\ &= -\mathbf{V}_+\Lambda_+^{-1}\mathbf{V}_+^T\mathbf{g} - \mathbf{V}_-\Lambda_-^{-1}\mathbf{V}_-^T\mathbf{g} \\ &= \underbrace{-\mathbf{V}_+\Lambda_+^{-1}\mathbf{V}_+^T\mathbf{g}}_{\mathbf{p}_+^N} - \underbrace{-\mathbf{V}_-\Lambda_-^{-1}\mathbf{V}_-^T\mathbf{g}}_{\mathbf{p}_-^N} \stackrel{\text{def}}{=} \mathbf{p}_+^N + \mathbf{p}_-^N \end{aligned}$$

where

$$\mathbf{p}_+^N = -\mathbf{V}_+\Lambda_+^{-1}\mathbf{V}_+^T\mathbf{g} \quad \text{and} \quad \mathbf{p}_-^N = -\mathbf{V}_-\Lambda_-^{-1}\mathbf{V}_-^T\mathbf{g}$$

21/106

22/106

Result

Given $\mathbf{m}_k^Q(\mathbf{p}) = \mathbf{g}^T\mathbf{p} + \frac{1}{2}\mathbf{p}^T\mathbf{H}\mathbf{p}$ with \mathbf{H} nonsingular, the Newton direction

$$\mathbf{p}^N = \mathbf{p}_+^N + \mathbf{p}_-^N = -\mathbf{V}_+\Lambda_+^{-1}\mathbf{V}_+^T\mathbf{g} - \mathbf{V}_-\Lambda_-^{-1}\mathbf{V}_-^T\mathbf{g}$$

is such that

- \mathbf{p}_+^N minimizes $\mathbf{m}_k^Q(\mathbf{p})$ on the subspace spanned by \mathbf{V}_+ .
- \mathbf{p}_-^N maximizes $\mathbf{m}_k^Q(\mathbf{p})$ on the subspace spanned by \mathbf{V}_- .

Proof:

- The function $\psi(y) \stackrel{\text{def}}{=} \mathbf{m}_k^Q(\mathbf{V}_+y)$ may be written as

$$\psi(y) = \mathbf{g}^T\mathbf{V}_+y + \frac{1}{2}y^T\mathbf{V}_+\mathbf{H}\mathbf{V}_+y$$

and has a stationary point y^* given by

$$y^* = -(\mathbf{V}_+\mathbf{H}\mathbf{V}_+)^{-1}\mathbf{V}_+\mathbf{g}$$

Since

$$\nabla^2\psi(y) = \mathbf{V}_+\mathbf{H}\mathbf{V}_+ = \mathbf{V}_+\mathbf{V}\Lambda\mathbf{V}^T\mathbf{V}_+ = \Lambda_+ \succ \mathbf{0}$$

we know that y^* is, in fact, the **unique minimizer** of $\psi(y)$ and

$$\mathbf{V}_+y^* = -\mathbf{V}_+\Lambda_+^{-1}\mathbf{V}_+\mathbf{g} \equiv \mathbf{p}_+^N$$

- For the second part

$$y^* = -(\mathbf{V}_-\mathbf{H}\mathbf{V}_-)^{-1}\mathbf{V}_-\mathbf{g} = -\Lambda_-^{-1}\mathbf{V}_-\mathbf{g}$$

so that

$$\mathbf{V}_-y^* = -\mathbf{V}_-\Lambda_-^{-1}\mathbf{V}_-\mathbf{g} \equiv \mathbf{p}_-^N.$$

Since $\Lambda_- \prec \mathbf{0}$, we know that \mathbf{p}_-^N **maximizes** $\mathbf{m}_k^Q(\mathbf{p})$ on the space spanned by the columns of \mathbf{V}_- . ■

Some observations

- \mathbf{p}_+^N is a **descent direction** for f :

$$\mathbf{g}^T\mathbf{p}_+^N = -\mathbf{g}^T\mathbf{V}_+\Lambda_+^{-1}\mathbf{V}_+\mathbf{g} = -y^T\Lambda_+^{-1}y \leq 0$$

where $y \stackrel{\text{def}}{=} \mathbf{V}_+\mathbf{g}$. The inequality is strict if $y \neq \mathbf{0}$, i.e., \mathbf{g} is not orthogonal to the column space of \mathbf{V}_+

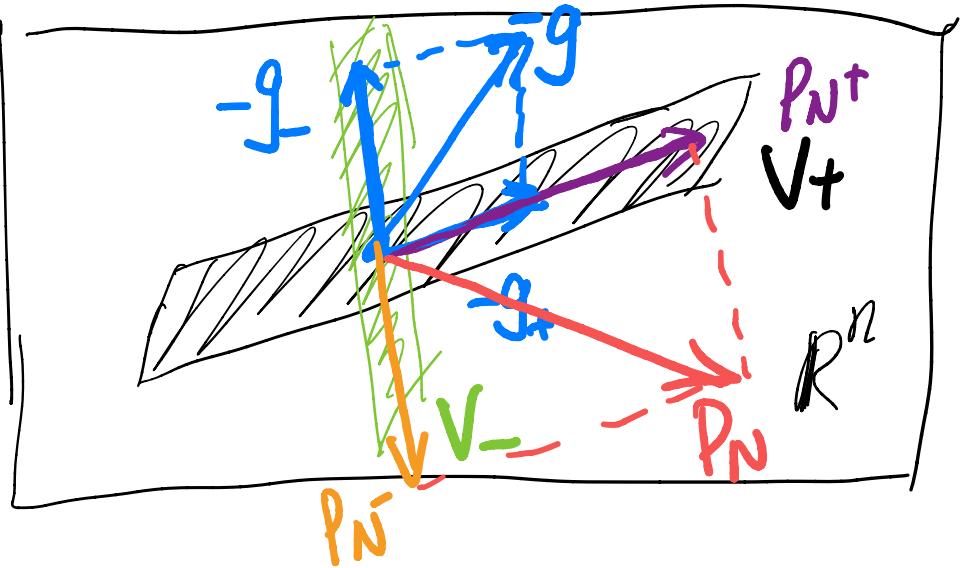
- \mathbf{p}_-^N is an **ascent direction** for f :

$$\mathbf{g}^T\mathbf{p}_-^N = -\mathbf{g}^T\mathbf{V}_-\Lambda_-^{-1}\mathbf{V}_-\mathbf{g} = -y^T\Lambda_-^{-1}y \geq 0$$

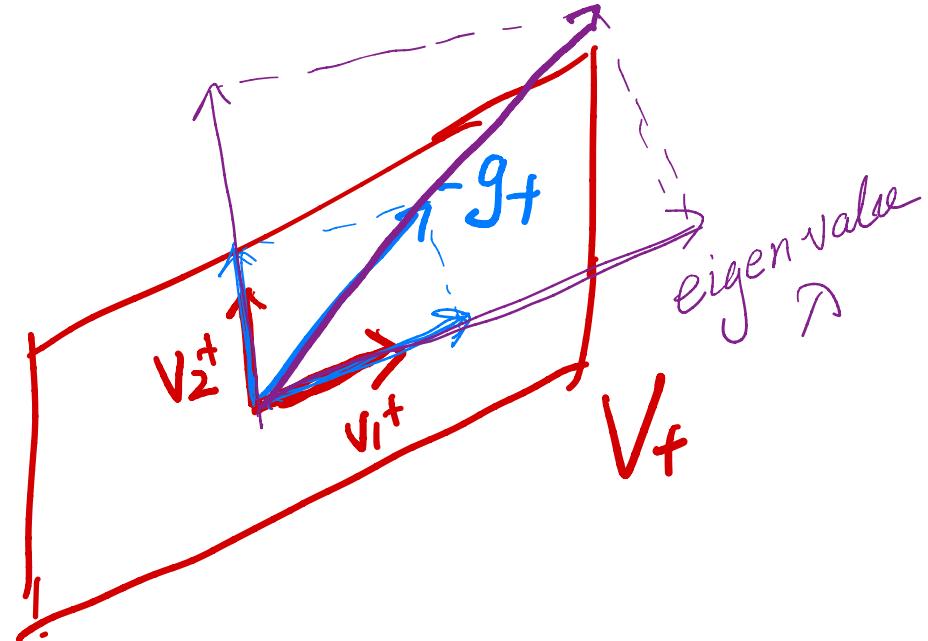
where $y \stackrel{\text{def}}{=} \mathbf{V}_-\mathbf{g}$. The inequality is strict if $y \neq \mathbf{0}$, i.e., \mathbf{g} is not orthogonal to the column space of \mathbf{V}_-

- $\mathbf{p}^N = \mathbf{p}_+^N + \mathbf{p}_-^N$ may or may not be a descent direction!

- This is the geometry of the Newton direction. What about the modified Newton direction?



V_+



$$P_N = P_N^+ + P_N^- \\ = -V_+ \Lambda_+^{-1} V_+^T g - V_- \Lambda_-^{-1} V_-^T g$$

P_N^+ is the point in V_f where quadratic $g_+^T P + \frac{1}{2} P^T H_+ P$ have gradient 0

$H_+ = V_+ \Lambda_+ V_+^T > 0 \Rightarrow$ minimizer (Convex function)

P_N^- is the point in V_- where quadratic $g_-^T P + \frac{1}{2} P^T H_- P$ have gradient 0

$H_- = V_- \Lambda_- V_-^T < 0 \Rightarrow$ maximizer (Concave function)

Question: what is the geometry of the modified-Newton step?

Answer: Partition the matrices in the spectral decomposition $H = V\Lambda V^T$ such that

$$V = [V_+ \ V_\epsilon \ V_-] \quad \text{and} \quad \Lambda = \begin{pmatrix} \Lambda_+ & & \\ & \Lambda_\epsilon & \\ & & \Lambda_- \end{pmatrix}$$

where

$$\Lambda_+ = \{\lambda_i : \lambda_i \geq \epsilon\} \quad \Lambda_\epsilon = \{\lambda_i : |\lambda_i| < \epsilon\} \quad \Lambda_- = \{\lambda_i : \lambda_i \leq -\epsilon\}$$

Theorem 2.3 (Properties of the modified-Newton direction)

Suppose that the positive-definite matrix B_k is computed from Algorithm 2 with input H_k and value $\epsilon > 0$. If $g_k \neq 0$ and p_k is the unique solution to $B_k p = -g_k$, then p_k may be written as $p_k = p_+ + p_\epsilon + p_-$, where

- (i) p_+ is a direction of positive curvature that minimizes $m_k^\varrho(p)$ in the space spanned by the columns of V_+ ;
- (ii) $-(p_-)$ is a direction of negative curvature that maximizes $m_k^\varrho(p)$ in the space spanned by the columns of V_- ; and
- (iii) p_ϵ is a multiple of the steepest-descent direction in the space spanned by the columns of V_ϵ with $\|p_\epsilon\|_2 = O(1/\epsilon)$.

Comments:

- implies that singular Hessians H_k may easily cause numerical difficulties
- the direction p_- is the negative of p_+ associated with the Newton step

25 / 106

The optimization problem solved in the $(k+1)$ st iterate will be

$$\underset{p \in \mathbb{R}^n}{\text{minimize}} \quad m_{k+1}^\varrho(p) \stackrel{\text{def}}{=} f_{k+1} + g_{k+1}^T p + \frac{1}{2} p^T B_{k+1} p \rightarrow \text{convex}$$

How do we choose B_{k+1} ?

How about to satisfy the following:

- the model should agree with f at x_{k+1}
- the gradient of the model should agree with the gradient of f at x_{k+1} $B_{k+1} = 0$
- the gradient of the model should agree with the gradient of f at x_k (previous point)

These are equivalent to

$$① \quad m_{k+1}^\varrho(0) = f_{k+1} \quad \checkmark \quad \text{already satisfied}$$

$$② \quad \nabla m_{k+1}^\varrho(0) = g_{k+1} \quad \checkmark \quad \text{already satisfied}$$

$$③ \quad \nabla m_{k+1}^\varrho(-\alpha_k p_k) = g_k \quad x_{k+1} - x_k = \alpha_k p_k$$

$$B_{k+1}(-\alpha_k p_k) + g_{k+1} = g_k \iff B_{k+1} \underbrace{(x_{k+1} - x_k)}_{s_k} = \underbrace{g_{k+1} - g_k}_{y_k}$$

Thus, we aim to cheaply compute a symmetric positive-definite matrix B_{k+1} that satisfies the so-called secant equation

$$B_{k+1} s_k = y_k \quad \text{where } s_k \stackrel{\text{def}}{=} x_{k+1} - x_k \text{ and } y_k \stackrel{\text{def}}{=} g_{k+1} - g_k$$

Main goals of quasi-Newton methods

- Maintain positive-definite “approximation” B_k to H_k
- Instead of computing B_{k+1} from scratch, e.g., from a spectral decomposition of H_{k+1} , want to compute B_{k+1} as an update to B_k using information at x_{k+1}
- inject “real” curvature information from $H(x_{k+1})$ into B_{k+1}
- Should be cheap to form and compute with B_k
- be applicable for medium-scale problems ($n \approx 1000 - 10,000$)
- Hope that fast convergence (superlinear) of the iterates $\{x_k\}$ computed using B_k can be recovered

28 / 106

27 / 106

Observation: If the matrix B_{k+1} is positive definite, then the curvature condition

$$s_k^T y_k > 0$$

is satisfied.

- using the fact that $B_{k+1} s_k = y_k$, we see that

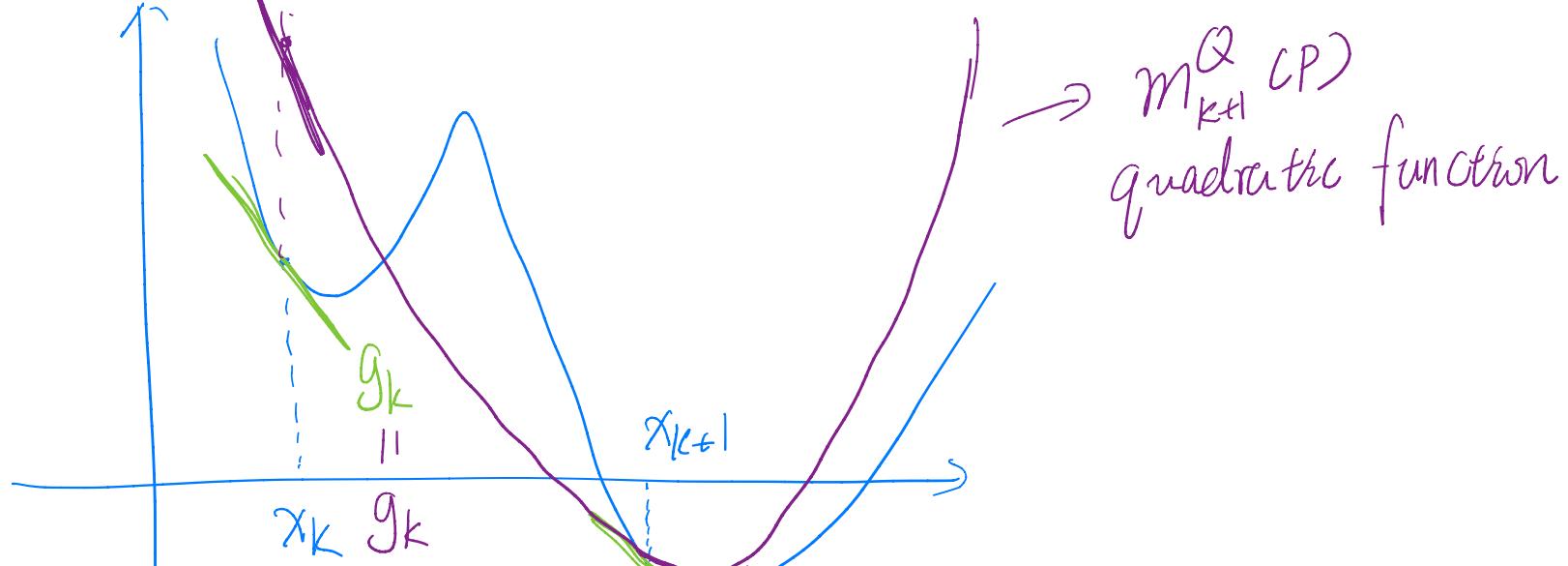
$$s_k^T y_k = s_k^T B_{k+1} s_k$$

which must be positive if $B_{k+1} \succ 0$

- the previous relation explains why (1) is called a curvature condition
- (1) will hold for any two points x_k and x_{k+1} if f is strictly convex
- (1) does not always hold for any two points x_k and x_{k+1} if f is nonconvex
- for nonconvex functions, we need to be careful how we choose α_k that defines $x_{k+1} = x_k + \alpha_k p_k$ (see Wolfe conditions for linesearch strategies)

Note: for the remainder, I will drop the subscript k from all terms.

29 / 106



violation case

$$x_k \xrightarrow{sk} x_{k+1}$$

$g_k > g_{k+1} \rightarrow$ can not fit a convex function into this scenario

$m_{k+1}^Q(x)$
quadratic function

$$s_k = x_{k+1} - x_k$$

$$y_k = g_{k+1} - g_k$$

Secant condition

$$s_k^T y_k > 0$$

$$s_k^T (g_{k+1} - g_k)$$

$$= s_k^T g_{k+1} - s_k^T g_k > 0$$

Gradient increase in s_k direction (Right)

Condition to
fit a convex function

The problem

Given a symmetric positive-definite matrix B , and vectors s and y , find a symmetric positive-definite matrix \bar{B} that satisfies the secant equation

$$\bar{B}s = y$$

$$s_k = \gamma_{k+1} - \gamma_k$$

$$g_k = g_{k+1} - g_k$$

Want a cheap update, so let us first consider a rank-one update of the form

$$\bar{B} = B + uv^T$$

for some vectors u and v that we seek.

Derivation:

Case 1: $Bs = y$

Choose $u = v = \mathbf{0}$ since

$$\bar{B}s = (B + uv^T)s = Bs + uv^Ts = y$$

Case 2: $Bs \neq y$

If $u = \mathbf{0}$ or $v = \mathbf{0}$ then

$$\bar{B}s = (B + uv^T)s = Bs + uv^Ts = Bs \neq y$$

So search for $u \neq \mathbf{0}$ and $v \neq \mathbf{0}$. Symmetry of B and desired symmetry of \bar{B} imply that

$$\bar{B} = \bar{B}^T \iff B + uv^T = B^T + vu^T \iff uv^T = vu^T$$

But $uv^T = vu^T$, $u \neq \mathbf{0}$, and $v \neq \mathbf{0}$ imply that $v = \alpha u$ for some $\alpha \neq 0$ (exercise). Thus, we should search for an $\alpha \neq 0$ and $u \neq \mathbf{0}$ such that the secant condition is satisfied by

$$\bar{B} = B + \alpha uu^T$$

find α and u to
make \bar{B} satisfy secant
condition

30/106

Definition 2.4 (SR1 quasi-Newton formula)

If $(y - Bs)^Ts \neq 0$ for some symmetric matrix B and vectors s and y , then the symmetric rank-1 (SR1) quasi-Newton formula for updating B is

$$\bar{B} = B + \frac{(y - Bs)(y - Bs)^T}{(y - Bs)^Ts}$$

could be negative

and satisfies

- \bar{B} is symmetric
- $\bar{B}s = y$ (secant condition)

In practice, we choose some $\kappa \in (0, 1)$ and then use

$$\bar{B} = \begin{cases} B + \frac{(y - Bs)(y - Bs)^T}{(y - Bs)^Ts} & \text{if } |s^T(y - Bs)| > \kappa \|s\|_2 \|y - Bs\|_2 \\ B & \text{otherwise} \end{cases}$$

to avoid numerical error.

Question: Is \bar{B} positive definite if B is positive definite?

Answer: Sometimes...but sometimes not!

From previous slide, we hope to find an $\alpha > 0$ and $u \neq \mathbf{0}$ so that

$$\bar{B} = B + \alpha uu^T$$

satisfies

$$y \stackrel{\text{want}}{=} \bar{B}s = Bs + \alpha uu^Ts = Bs + \alpha(u^Ts)u$$

which implies

$$\alpha(u^Ts)u = y - Bs \neq 0.$$

This implies that

$$u = \beta(y - Bs) \quad \text{for some } \beta \neq 0.$$

Plugging back in, we are searching for

$$\bar{B} = B + \alpha\beta^2(y - Bs)(y - Bs)^T$$

$$= B + \gamma(y - Bs)(y - Bs)^T \quad \text{for some } \gamma \neq 0$$

that satisfies the secant equation

$$y \stackrel{\text{want}}{=} \bar{B}s = Bs + \gamma(y - Bs)^Ts(y - Bs)$$

which implies

$$y - Bs = \gamma(y - Bs)^Ts(y - Bs)$$

which means we can choose

$$\gamma = \frac{1}{(y - Bs)^Ts} \quad \text{provided } (y - Bs)^Ts \neq 0.$$

31/106

Example 2.5 (SR1 update is not necessarily positive definite)

Consider the SR1 update with

$$B = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \quad s = \begin{pmatrix} -1 \\ -1 \end{pmatrix} \quad \text{and} \quad y = \begin{pmatrix} -3 \\ 2 \end{pmatrix}$$

The matrix B is positive definite with eigenvalues $\approx \{0.38, 2.6\}$. Moreover,

$$y^Ts = (-3 \ 2) \begin{pmatrix} -1 \\ -1 \end{pmatrix} = 1 > 0$$

so that the necessary condition for \bar{B} to be positive definite holds. The SR1 update gives

$$y - Bs = \begin{pmatrix} -3 \\ 2 \end{pmatrix} - \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} -1 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 4 \end{pmatrix}$$

$$(y - Bs)^Ts = (0 \ 4) \begin{pmatrix} -1 \\ -1 \end{pmatrix} = -4$$

$$\bar{B} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} + \frac{1}{(-4)} \begin{pmatrix} 0 \\ 4 \end{pmatrix} \begin{pmatrix} 0 & 4 \end{pmatrix} = \begin{pmatrix} 2 & -1 \\ 1 & -3 \end{pmatrix}$$

The matrix \bar{B} is indefinite with eigenvalues $\approx \{2.19, -3.19\}$.

33/106

Theorem 2.6 (convergence of SR1 update)

Suppose that

- f is C^2 on \mathbb{R}^n
- $\lim_{k \rightarrow \infty} x_k = x^*$ for some vector x^*
- $\nabla^2 f$ is bounded and Lipschitz continuous in a neighborhood of x^*
- the steps s_k are uniformly linearly independent
- $|s_k^T(y_k - B_k s_k)| > \kappa \|s_k\|_2 \|y_k - B_k s_k\|_2$ for some $\kappa \in (0, 1)$

Then, the matrices generated by the SR1 formula satisfy

$$\lim_{k \rightarrow \infty} B_k = \nabla^2 f(x^*)$$

Comment: the SR1 updates may converge to an **indefinite** matrix

line search method require positive definite matrix B_k
 to satisfy secant condition, but not for trust region method.
 Because trust region method can work with negative B_k

34 / 106

35 / 106

For **hereditary positive definiteness** we must consider updates of at least rank two.

Result

U is a symmetric matrix of rank two if and only if

$$U = \beta uu^T + \delta ww^T,$$

for some nonzero β and δ , with u and w linearly independent.

If

$$\bar{B} = B + U$$

for some **rank-two** matrix U and $\bar{B}s = y$ for some vectors s and y , then

$$\bar{B}s = (B + U)s = y,$$

and it follows that

$$y - Bs = Us = \beta(u^T s)u + \delta(w^T s)w$$

Some comments

- if B is symmetric, then the SR1 update \bar{B} (when it exists) is symmetric
- if B is positive definite, then the SR1 update \bar{B} (when it exists) is **not necessarily** positive definite
- linesearch methods require **positive-definite** matrices
- SR1 updating is **not** appropriate (in general) for linesearch methods
- SR1 updating is an **attractive** option for optimization methods that effectively utilize **indefinite** matrices (e.g., trust-region methods)
- to derive an updating strategy that produces **positive-definite** matrices, we have to look **beyond rank-one** updates. Would a **rank-two** update work?

Let us make the “obvious” choices of

$$u = Bs \quad \text{and} \quad w = y$$

which gives the rank-two update

$$U = \beta Bs(Bs)^T + \delta yy^T$$

and we now search for β and δ so that $\bar{B}s = y$ is satisfied. Substituting, we have

$$\begin{aligned} y &\stackrel{\text{want}}{=} \bar{B}s = (B + \beta Bs(Bs)^T + \delta yy^T)s \\ &= Bs + \beta Bss^T Bs + \delta yy^Ts \\ &= Bs + \beta(s^T Bs)Bs + \delta(y^T s)y \end{aligned}$$

Making the “simple” choice of

$$\beta = -\frac{1}{s^T Bs} \quad \text{and} \quad \delta = \frac{1}{y^T s}$$

54:37

gives

$$\begin{aligned} \bar{B}s &= Bs - \frac{1}{s^T Bs}(s^T Bs)Bs + \frac{1}{y^T s}(y^T s)y \\ &= Bs - Bs + y = y \end{aligned}$$

as desired. This gives the Broyden, Fletcher, Goldfarb, Shanno (**BFGS**) quasi-Newton update formula.

Broyden, Fletcher, Goldfarb, Shanno (BFGS) update

Given a symmetric positive-definite matrix B and vectors s and y that satisfy $y^T s > 0$, the quasi-Newton **BFGS update formula** is given by

$$\bar{B} = B - \frac{1}{s^T B s} B s (B s)^T + \frac{1}{y^T s} y y^T \quad (2)$$

Result

If B is positive definite and $y^T s > 0$, then

$$\bar{B} = B - \frac{1}{s^T B s} B s (B s)^T + \frac{1}{y^T s} y y^T$$

is positive definite and satisfies $\bar{B}s = y$.

Proof:

Homework assignment. (hint: see the next slide)

38 / 106

Recall: given the positive-definite matrix B and vectors s and y satisfying $s^T y > 0$, the BFGS formula (2) may be written as

$$\bar{B} = B - aa^T + bb^T$$

advantage: trace x, y, g vector instead of $n \times n$ B_p matrix

where

$$a = \frac{Bs}{(s^T Bs)^{1/2}} \quad \text{and} \quad b = \frac{y}{(y^T s)^{1/2}}$$

Limited-memory BFGS update

Suppose at iteration k we have m previous pairs (s_i, y_i) for $i = k - m, \dots, k - 1$. Then the limited-memory BFGS (L-BFGS) update with **memory m** and **positive-definite seed matrix B_k^0** may be written in the form

$$B_k = B_k^0 + \sum_{i=k-m}^{k-1} [b_i b_i^T - a_i a_i^T]$$

for some vectors a_i and b_i that may be recovered via an **unrolling formula** (next slide).

- the positive-definite seed matrix B_k^0 is often chosen to be diagonal
- essentially perform m BFGS updates to the seed matrix B_k^0
- A different seed matrix may be used for each iteration k
 - ▶ Added flexibility allows “better” seed matrices to be used as iterations proceed

41 / 106

A second derivation of the BFGS formula

Let $M = B^{-1}$ for some given symmetric positive definite matrix B . Solve

$$\underset{\bar{M} \in \mathbb{R}^{n \times n}}{\text{minimize}} \|\bar{M} - M\|_W \quad \text{subject to} \quad \bar{M} = \bar{M}^T, \quad \bar{M}y = s$$

for a “closest” symmetric matrix \bar{M} , where

$$\|A\|_W = \|W^{1/2} A W^{1/2}\|_F \quad \text{and} \quad \|C\|_F = \sum_{i=1}^n \sum_{j=1}^n c_{ij}^2$$

for any weighting matrix W that satisfies $Wy = s$. If we choose

$$W = G^{-1} \quad \text{and} \quad G = \int_0^1 [\nabla^2 f(y + \tau s)]^{-1} d\tau$$

then the solution is

$$\bar{M} = \left(I - \frac{sy^T}{y^T s}\right) M \left(I - \frac{ys^T}{y^T s}\right) + \frac{ss^T}{y^T s}. \quad \text{→ positive definite}$$

Computing the inverse $\bar{B} = \bar{M}^{-1}$ using the Sherman-Morrison-Woodbury formula gives

$$\bar{B} = B - \frac{1}{s^T B s} B s (B s)^T + \frac{1}{y^T s} y y^T \quad (\text{BFGS formula again})$$

so that $\bar{B} \succ 0$ if and only if $\bar{M} \succ 0$.

39 / 106

Algorithm 3 Unrolling the L-BFGS formula

```

1: input  $m \geq 1$ ,  $\{(s_i, y_i)\}$  for  $i = k - m, \dots, k - 1$ , and  $B_k^0 \succ 0$ 
2: for  $i = k - m, \dots, k - 1$  do
3:    $b_i \leftarrow y_i / (y_i^T s_i)^{1/2}$ 
4:    $a_i \leftarrow B_k^0 s_i + \sum_{j=k-m}^{i-1} [(b_j^T s_i) b_j - (a_j^T s_i) a_j]$ 
5:    $a_i \leftarrow a_i / (s_i^T a_i)^{1/2}$ 
6: end for

```

seed matrix

- typically $3 \leq m \leq 30$
- b_i and $b_j^T s_i$ should be saved and reused from previous iterations (exceptions are b_{k-1} and $b_j^T s_{k-1}$ because they depend on the most recent data obtained).
- with the previous savings in computation and assuming $B_k^0 = I$, the total cost to get the a_i, b_i vectors is approximately $\frac{3}{2}m^2n$.
- cost of a matrix-vector product with B_k requires $4mn$ multiplications.
- other so-called **compact representations** are slightly more efficient

42 / 106

The problem of interest

Given a **symmetric positive-definite** matrix A , solve the linear system

$$Ax = b$$

which is equivalent to finding the **unique minimizer** of

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad q(x) \stackrel{\text{def}}{=} \frac{1}{2}x^T Ax - b^T x \quad (3)$$

because $\nabla q(x) = Ax - b$ and $\nabla^2 q(x) = A \succ 0$ implies that

$$\nabla q(x) = Ax - b = 0 \iff Ax = b$$

- We seek a method that cheaply computes **approximate** solutions to the system $Ax = b$, or equivalent approximately minimizes (3).
- Do not want to compute a factorization of A , because it is assumed to be too expensive. We are interested in very large-scale problems
- We allow ourselves to compute matrix-vector products, i.e., Av for any vector v
- Why do we care about this? We will see soon!

Notation: $r(x) \stackrel{\text{def}}{=} Ax - b$ and $r_k \stackrel{\text{def}}{=} Ax_k - b$.

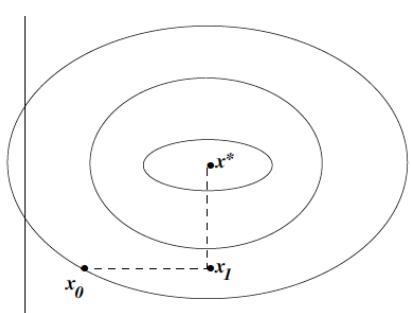


Figure: Coordinate descent converges in n steps. Very good! Axes are aligned with the coordinate directions e_i .

Fact: the axes are determined by the eigenvectors of A .

Observation: coordinate descent converges in no more than n steps if the eigenvectors align with the coordinate directions, i.e., $V = I$ where V is the eigenvector matrix of A .

Implication: Using the spectral decomposition of A

$$A = V\Lambda V^T = I\Lambda I^T = \Lambda \quad (\text{a diagonal matrix})$$

Conclusion: Coordinate descent converges in $\leq n$ steps when A is **diagonal**.

44 / 106

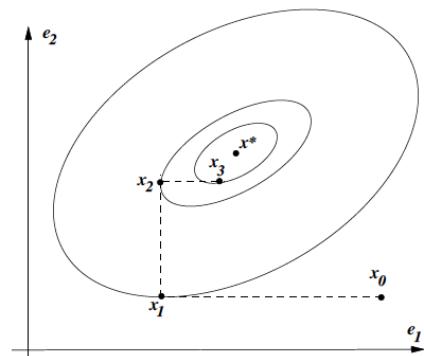


Figure: Coordinate descent does not converge in n steps. Looks like it will be very slow! Axes are not aligned with the coordinate directions e_i .

46 / 106

$$q(x) = \frac{1}{2}x^T Ax - b^T x$$

Algorithm 4 Coordinate descent algorithm for minimizing q

```

1: input initial guess  $x_0 \in \mathbb{R}^n$  and a symmetric positive-definite matrix  $A \in \mathbb{R}^{n \times n}$ 
2: loop
3:   for  $i = 0, 1, \dots, n - 1$  do
4:     Compute  $\alpha_i$  as the solution to
      
$$\underset{\alpha \in \mathbb{R}}{\text{minimize}} \quad q(x_i + \alpha e_{i+1})$$

5:      $x_{i+1} \leftarrow x_i + \alpha_i e_{i+1}$ 
6:   end for
7:    $x_0 \leftarrow x_n$ 
8: end loop

```

- e_j represents the j th coordinate vector, i.e.,

$$e_j = (0 \ 0 \ \dots \ \overset{j\text{th}}{\overbrace{1}} \ \dots \ 0 \ 0)^T \in \mathbb{R}^n$$

- Is it easy to solve (4)? Yes! ([exercise](#))
- Does this work?

45 / 106

First thought: This is only good for diagonal matrices. That sucks!

Second thought: Can we use a transformation of variables so that the **transformed Hessian** is diagonal?

- Let S be a square nonsingular matrix and define the transformed variables

$$y = S^{-1}x \iff Sy = x.$$

Consider the transformed quadratic function \hat{q} defined as

$$\hat{q}(y) \stackrel{\text{def}}{=} q(Sy) \equiv \frac{1}{2}y^T S^T AS y - (S^T b)^T y = \frac{1}{2}y^T \hat{A} y - \hat{b}^T y$$

so that a minimizer x^* of q satisfies $x^* = Sy^*$ where y^* is a minimizer of \hat{q} .

- If $\hat{A} = S^T AS$ is diagonal, then applying the coordinate descent Algorithm 4 to \hat{q} would converge to y^* in $\leq n$ steps.
- \hat{A} is diagonal if and only if

$$s_i^T \hat{A} s_j = 0 \quad \text{for all } i \neq j,$$

where

$$S = (s_0 \ s_1 \ \dots \ s_{n-1})$$

- Line 5 of Algorithm 4 (in the transformed variables) is

$$y_{i+1} = y_i + \alpha_i e_{i+1}.$$

Multiplying by S and using $Sy = x$ leads to

$$x_{i+1} = Sy_{i+1} = S(y_i + \alpha_i e_{i+1}) = Sy_i + \alpha_i Se_{i+1} = x_i + \alpha_i s_i$$

47 / 106

Definition 2.7 (A -conjugate directions)

A set of nonzero vectors $\{s_0, s_1, \dots, s_l\}$ is said to be conjugate with respect to the symmetric positive-definite matrix A (**A -conjugate** for short) if

$$s_i^T A s_j = \mathbf{0} \quad \text{for all } i \neq j.$$

- We want to find A -conjugate vectors.
- The eigenvectors of A are A -conjugate since the spectral decomposition gives

$$A = V \Lambda V^T \iff V^T A V = \Lambda \quad (\text{diagonal matrix of eigenvalues})$$

- Computing the spectral decomposition is **too expensive** for **large scale** problems.
- We want to find A -conjugate vectors **cheaply**!
- Let us look at a general algorithm.

Exercise: show that the α_k that solves the minimization problem on line 5 of Algorithm 5 satisfies

$$\alpha_k = -\frac{s_k^T r_k}{s_k^T A s_k}.$$

Before discussing **how** we compute the A -conjugate directions $\{s_i\}$, let us look at some properties of them if we assume we already have them.

Theorem 2.9 (Expanding subspace minimization)

Let $x_0 \in \mathbb{R}^n$ be any starting point and suppose that $\{x_k\}$ is the sequence of iterates generated by the A -conjugate direction Algorithm 5 for a set of A -conjugate directions $\{s_k\}$. Then,

$$r_k^T s_i = \mathbf{0} \quad \text{for all } i = 0, \dots, k-1$$

and x_k is the unique minimizer of $q(x) = \frac{1}{2} x^T A x - b^T x$ over the set

$$\{x : x = x_0 + \text{span}(s_0, s_1, \dots, s_{k-1})\}$$

Proof: See Theorem 5.2 in Nocedal and Wright.

Algorithm 5 General A -conjugate direction algorithm

```

1: input symmetric positive-definite matrix  $A \in \mathbb{R}^{n \times n}$ , and vectors  $x_0$  and  $b \in \mathbb{R}^n$ .
2: Set  $r_0 \leftarrow Ax_0 - b$  and  $k \leftarrow 0$ .
3: Choose direction  $s_0$ .
4: while  $r_k \neq \mathbf{0}$  do
5:   Compute  $\alpha_k \leftarrow \underset{\alpha \in \mathbb{R}}{\operatorname{argmin}} q(x_k + \alpha s_k)$ 
6:   Set  $x_{k+1} \leftarrow x_k + \alpha_k s_k$ .
7:   Set  $r_{k+1} \leftarrow Ax_{k+1} - b$ .
8:   Pick new  $A$ -conjugate direction  $s_{k+1}$ .
9:   Set  $k \leftarrow k + 1$ .
10: end while
```

- How do we compute α_k ?
- How do we compute the A -conjugate directions **efficiently**?

Theorem 2.8 (general A -conjugate algorithm converges)

For any $x_0 \in \mathbb{R}^n$, the general A -conjugate direction Algorithm 5 computes the solution x^* of the system $Ax = b$ in **at most n steps**.

Proof: Theorem 5.1 in Nocedal and Wright.

48 / 106

49 / 106

Question: If $\{s_i\}_{i=0}^{k-1}$ are A -conjugate, how do we compute s_k to be A -conjugate?

Answer: Theorem 2.9 shows that r_k is orthogonal to the previously explored space. To keep it **simple** and hopefully **cheap**, let us try

$$s_k = -r_k + \beta s_{k-1} \quad \text{for some } \beta. \tag{5}$$

If this is going to work, then the A -conjugacy condition and (5) requires that

$$\begin{aligned} \mathbf{0} &= s_{k-1}^T A s_k \\ &= s_{k-1}^T A (-r_k + \beta s_{k-1}) \\ &= s_{k-1}^T (-A r_k + \beta A s_{k-1}) \\ &= -s_{k-1}^T A r_k + \beta s_{k-1}^T A s_{k-1} \end{aligned}$$

and solving for β gives

$$\beta = \frac{s_{k-1}^T A r_k}{s_{k-1}^T A s_{k-1}}.$$

With this choice of β , the vector s_k is A -conjugate!....but, how do we get started? Choosing $s_0 = -r_0$, i.e., a steepest descent direction, makes sense.

This gives us a complete algorithm; the linear CG algorithm!

50 / 106

51 / 106

Algorithm 6 Preliminary linear CG algorithm

```

1: input symmetric positive-definite matrix  $A \in \mathbb{R}^{n \times n}$ , and vectors  $x_0$  and  $b \in \mathbb{R}^n$ .
2: Set  $r_0 \leftarrow Ax_0 - b$ ,  $s_0 \leftarrow -r_0$ , and  $k \leftarrow 0$ .
3: while  $r_k \neq 0$  do
4:   Set  $\alpha_k \leftarrow -(s_k^T r_k) / (s_k^T A s_k)$ .
5:   Set  $x_{k+1} \leftarrow x_k + \alpha_k s_k$ .
6:   Set  $r_{k+1} \leftarrow Ax_{k+1} - b$ .
7:   Set  $\beta_{k+1} \leftarrow (s_k^T A r_{k+1}) / (s_k^T A s_k)$ .
8:   Set  $s_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} s_k$ .
9:   Set  $k \leftarrow k + 1$ .
10: end while

```

- This **preliminary** version requires two matrix-vector products per iteration.
- By using some more “tricks” and algebra, we can reduce this to a **single** matrix-vector multiplication per iteration.

52/106

We want to use CG to compute an approximate solution p to

$$\underset{p \in \mathbb{R}^n}{\text{minimize}} \quad f + g^T p + \frac{1}{2} p^T H p$$

- H may be **indefinite** (even **singular**) and must ensure that p is a descent direction

Algorithm 8 Newton CG algorithm for computing descent directions

```

1: input symmetric matrix  $H \in \mathbb{R}^{n \times n}$  and vector  $g$ 
2: Set  $p_0 = 0$ ,  $r_0 \leftarrow g$ ,  $s_0 \leftarrow -g$ , and  $k \leftarrow 0$ .
3: while  $r_k \neq 0$  do
4:   if  $s_k^T H s_k > 0$  then
5:     Set  $\alpha_k \leftarrow (r_k^T r_k) / (s_k^T H s_k)$ .
6:   else
7:     if  $k = 0$ , then return  $p_k = -g$  else return  $p_k$  end if
8:   end if
9:   Set  $p_{k+1} \leftarrow p_k + \alpha_k s_k$ .
10:  Set  $r_{k+1} \leftarrow r_k + \alpha_k H s_k$ .
11:  Set  $\beta_{k+1} \leftarrow (r_{k+1}^T r_{k+1}) / (r_k^T r_k)$ .
12:  Set  $s_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} s_k$ .
13:  Set  $k \leftarrow k + 1$ .
14: end while
15: return  $p_k$ 

```

Made the replacements $x \leftarrow p$, $A \leftarrow H$, and $b \leftarrow -g$ in Algorithm 7.

54/106

Algorithm 7 Linear CG algorithm

```

1: input symmetric positive-definite matrix  $A \in \mathbb{R}^{n \times n}$ , and vectors  $x_0$  and  $b \in \mathbb{R}^n$ .
2: Set  $r_0 \leftarrow Ax_0 - b$ ,  $s_0 \leftarrow -r_0$ , and  $k \leftarrow 0$ .
3: while  $r_k \neq 0$  do
4:   Set  $\alpha_k \leftarrow (r_k^T r_k) / (s_k^T A s_k)$ .
5:   Set  $x_{k+1} \leftarrow x_k + \alpha_k s_k$ .
6:   Set  $r_{k+1} \leftarrow r_k + \alpha_k A s_k$ .
7:   Set  $\beta_{k+1} \leftarrow (r_{k+1}^T r_{k+1}) / (r_k^T r_k)$ .
8:   Set  $s_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} s_k$ .
9:   Set  $k \leftarrow k + 1$ .
10: end while

```

- Only requires the **single** matrix-vector multiplication As_k .
- More practical to use a stopping condition like

$$\|r_k\| \leq 10^{-8} \max(1, \|r_0\|_2)$$

- Required computation
 - ▶ 2 inner products
 - ▶ 3 vector sums
 - ▶ **1 matrix-vector multiplication**
- Ideal for large (sparse) matrices A
- Converges quickly if $\text{cond}(A)$ is small or the eigenvalues are “clustered”
 - ▶ convergence can be accelerated by using **preconditioning**

53/106

Theorem 2.10 (Newton CG algorithm computes descent directions.)

If $g \neq 0$ and Algorithm 8 terminates on iterate K , then every iterate p_k of the algorithm satisfies $p_k^T g < 0$ for $k \leq K$.

Proof:

Case 1: Algorithm 8 terminates on iteration $K = 0$.

It is immediate that $g^T p_0 = -g^T g = -\|g\|_2^2 < 0$ since $g \neq 0$.

Case 2: Algorithm 8 terminates on iteration $K \geq 1$.

First observe from $p_0 = 0$, $s_0 = -g$, $r_0 = g$, $g \neq 0$, and lines 10 and 9 of Algorithm 8 that

$$g^T p_1 = g^T(p_0 + \alpha_0 s_0) = \alpha_0 g^T s_0 = \frac{r_0^T r_0}{s_0^T H s_0} g^T s_0 = -\frac{\|g\|_2^4}{g^T H g} < 0. \quad (6)$$

By unrolling the “p” update, using A -conjugacy of $\{s_i\}$, and definition of r_k , we get

$$s_k^T r_k = s_k^T (H p_k + g) = s_k^T g + s_k^T H \sum_{j=0}^{k-1} \alpha_j s_j = s_k^T g.$$

The previous line, lines 9 and 5 of Algorithm 8, and the fact that $s_k^T r_k = -\|r_k\|_2^2$ gives

$$g^T p_{k+1} = g^T p_k + \alpha_k g^T s_k = g^T p_k + \alpha_k s_k^T r_k = g^T p_k + \frac{r_k^T r_k}{s_k^T H s_k} s_k^T r_k = g^T p_k - \frac{\|r_k\|_2^4}{s_k^T H s_k} < g^T p_k.$$

Combining the previous inequality and (6) results in

$$g^T p_{k+1} < g^T p_k < \dots < g^T p_1 < -\frac{\|g\|_2^4}{g^T H g} < 0. \blacksquare$$

55/106

To find α_k step length to get sufficient decrease.

Algorithm 9 Backtracking

```

1: input  $x_k, p_k$ 
2: Choose  $\alpha_{\text{init}} > 0$  and  $\tau \in (0, 1)$ 
3: Set  $\alpha^{(0)} = \alpha_{\text{init}}$  and  $l = 0$ 
4: until  $f(x_k + \alpha^{(l)} p_k) < f(x_k)$ 
5: Set  $\alpha^{(l+1)} = \tau \alpha^{(l)}$ 
6: Set  $l \leftarrow l + 1$ 
7: end until
8: return  $\alpha_k = \alpha^{(l)}$ 
```

- typical choices (not always)
 - $\alpha_{\text{init}} = 1$
 - $\tau = 1/2$
- this prevents the step from getting too small
- does not prevent the step from being too long
- decrease in f may not be proportional to the directional derivative
- need to tighten the requirement that

$$f(x_k + \alpha^{(l)} p_k) < f(x_k)$$

58 / 106

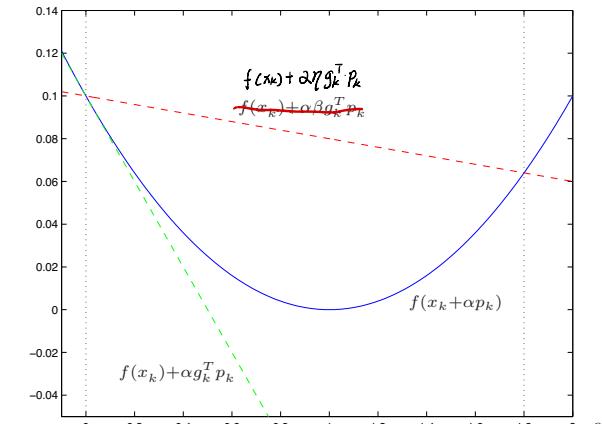
The Armijo condition

Given a point x_k and search direction p_k , we say that α_k satisfies the **Armijo condition** if

$$f(x_k + \alpha_k p_k) \leq f(x_k) + \eta \alpha_k g_k^T p_k \quad (7)$$

for some $\eta \in (0, 1)$, e.g., $\eta = 0.1$ or even $\eta = 0.0001$

Purpose: ensure the decrease in f is substantial relative to the directional derivative.



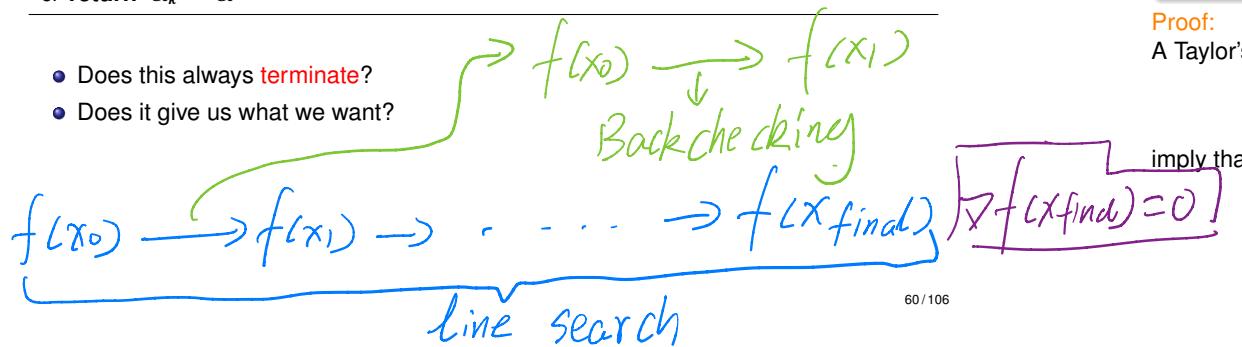
59 / 106

Algorithm 10 A backtracking-Armijo linesearch

```

1: input  $x_k, p_k$ 
2: Choose  $\alpha_{\text{init}} > 0, \eta \in (0, 1)$ , and  $\tau \in (0, 1)$ 
3: Set  $\alpha^{(0)} = \alpha_{\text{init}}$  and  $l = 0$ 
4: until  $f(x_k + \alpha^{(l)} p_k) \leq f(x_k) + \eta \alpha^{(l)} g_k^T p_k$ 
5: Set  $\alpha^{(l+1)} = \tau \alpha^{(l)}$ 
6: Set  $l \leftarrow l + 1$ 
7: end until
8: return  $\alpha_k = \alpha^{(l)}$ 
```

- Does this always terminate?
- Does it give us what we want?



60 / 106

Theorem 3.1 (Satisfying the Armijo condition)

Suppose that

- $f \in C^1$ and $g(x)$ is Lipschitz continuous with Lipschitz constant $\gamma(x)$
- p is a descent direction at x

Then for any given $\eta \in (0, 1)$, the Armijo condition

$$f(x + \alpha p) \leq f(x) + \eta \alpha g(x)^T p$$

is satisfied for all $\alpha \in [0, \alpha_{\max}(x)]$, where

$$\alpha_{\max}(x) := \frac{2(\eta - 1)g(x)^T p}{\gamma(x)\|p\|_2^2} > 0$$

Proof:

A Taylor's approximation and

$$\alpha \leq \frac{2(\eta - 1)g(x)^T p}{\gamma(x)\|p\|_2^2}$$

$$\begin{aligned}
 f(x + \alpha p) &\leq f(x) + \alpha g(x)^T p + \frac{1}{2}\gamma(x)\alpha^2\|p\|_2^2 \\
 &\leq f(x) + \alpha g(x)^T p + \alpha(\eta - 1)g(x)^T p \\
 &= f(x) + \alpha\eta g(x)^T p \quad \blacksquare
 \end{aligned}$$

61 / 106

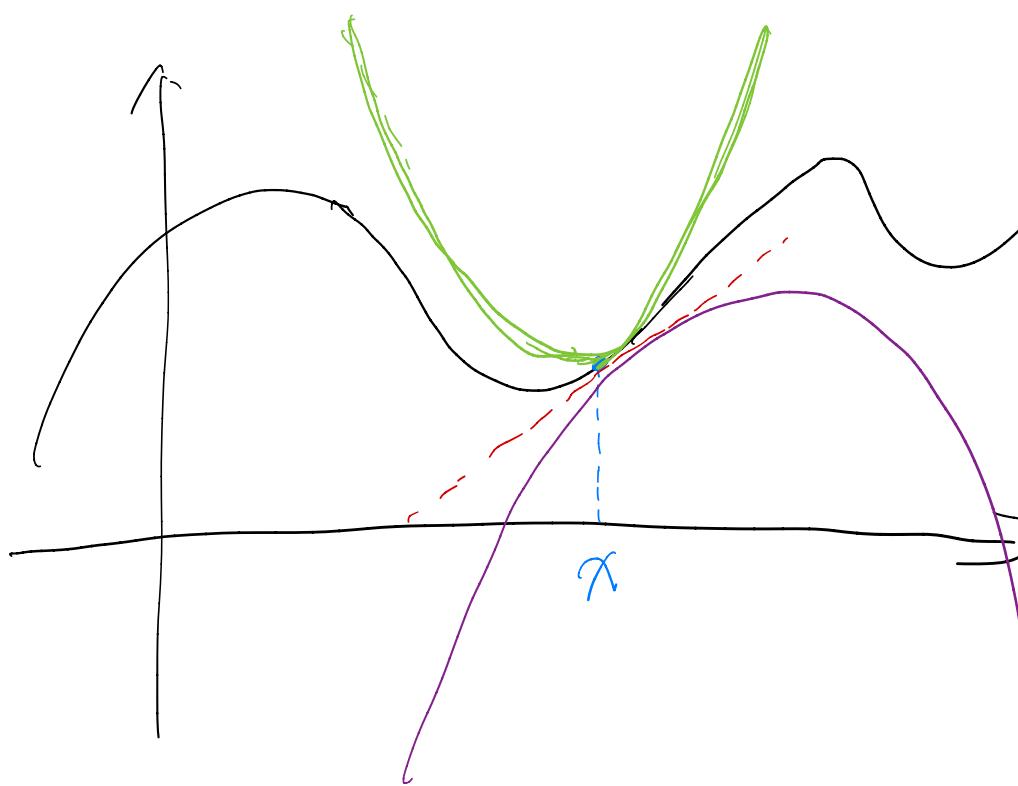
$f(x)$ continuously differentiable

$\nabla f(x)$ is lipschitz continuous
with constant $\gamma^L(x)$

Taylor: $|f(y) - f(x) - g(x)^T s| \leq \frac{1}{2} \gamma^L(x) \|s\|^2 \quad \forall s \in \mathbb{R}^n$
 $y = x + s$

$$\Rightarrow \boxed{f(x) + g(x)^T s - \frac{1}{2} \gamma^L(x) \|s\|^2} \quad f(y) \leq f(x) + g(x)^T s + \frac{1}{2} \gamma^L(x) \|s\|^2$$

quadratic function.

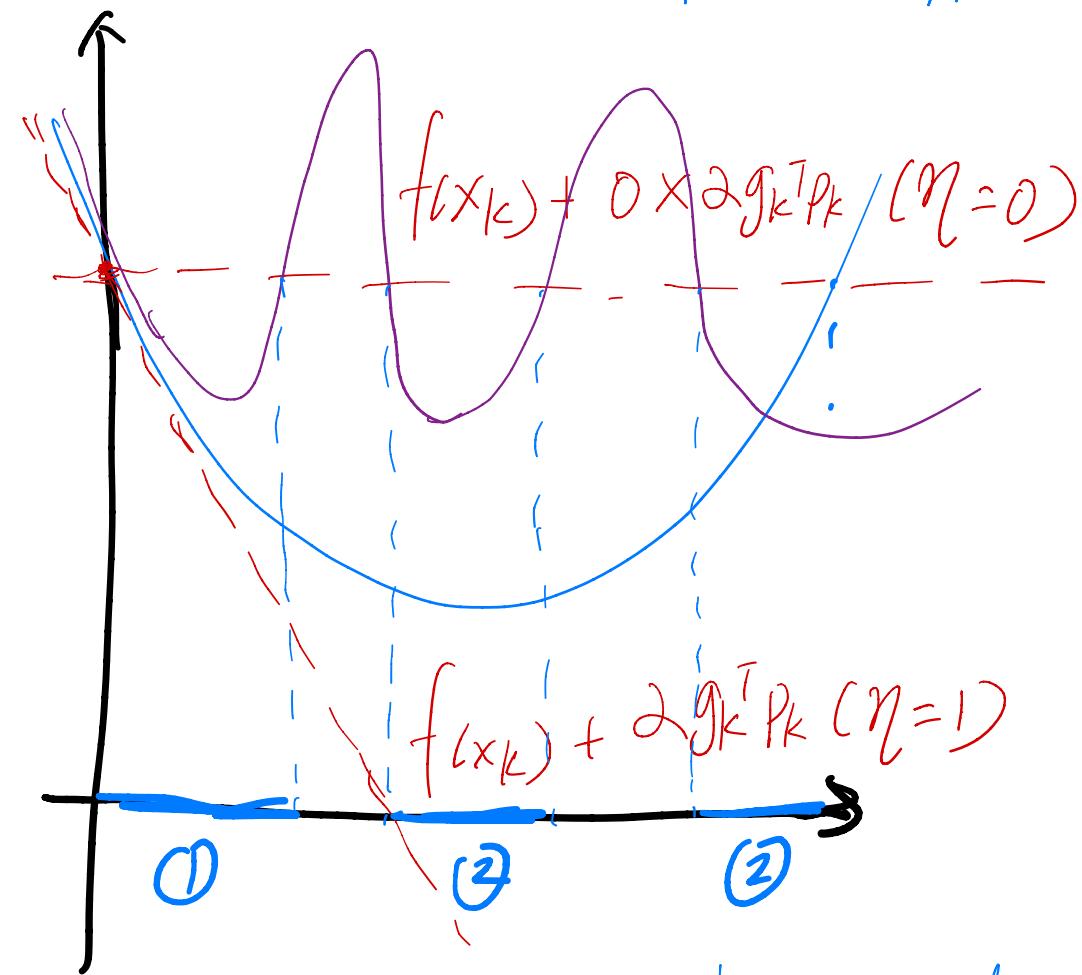


- Both $f(x)$ and quadratic function have same slope at point x

- These two function bound $f(y)$

Armijo Condition : we want a interval $[0, \Delta_{\max}]$ s.t
 $\forall \Delta \in [0, \Delta_{\max}]$ satisfy

$$f(x_k + \Delta_k p_k) \leq f(x_k) + \eta \Delta_k g(x_k)^T p_k$$



① ② ③ → These interval satisfy Armijo

$$f(x_k) + \underline{\eta \Delta_k g(x_k)^T p_k}$$

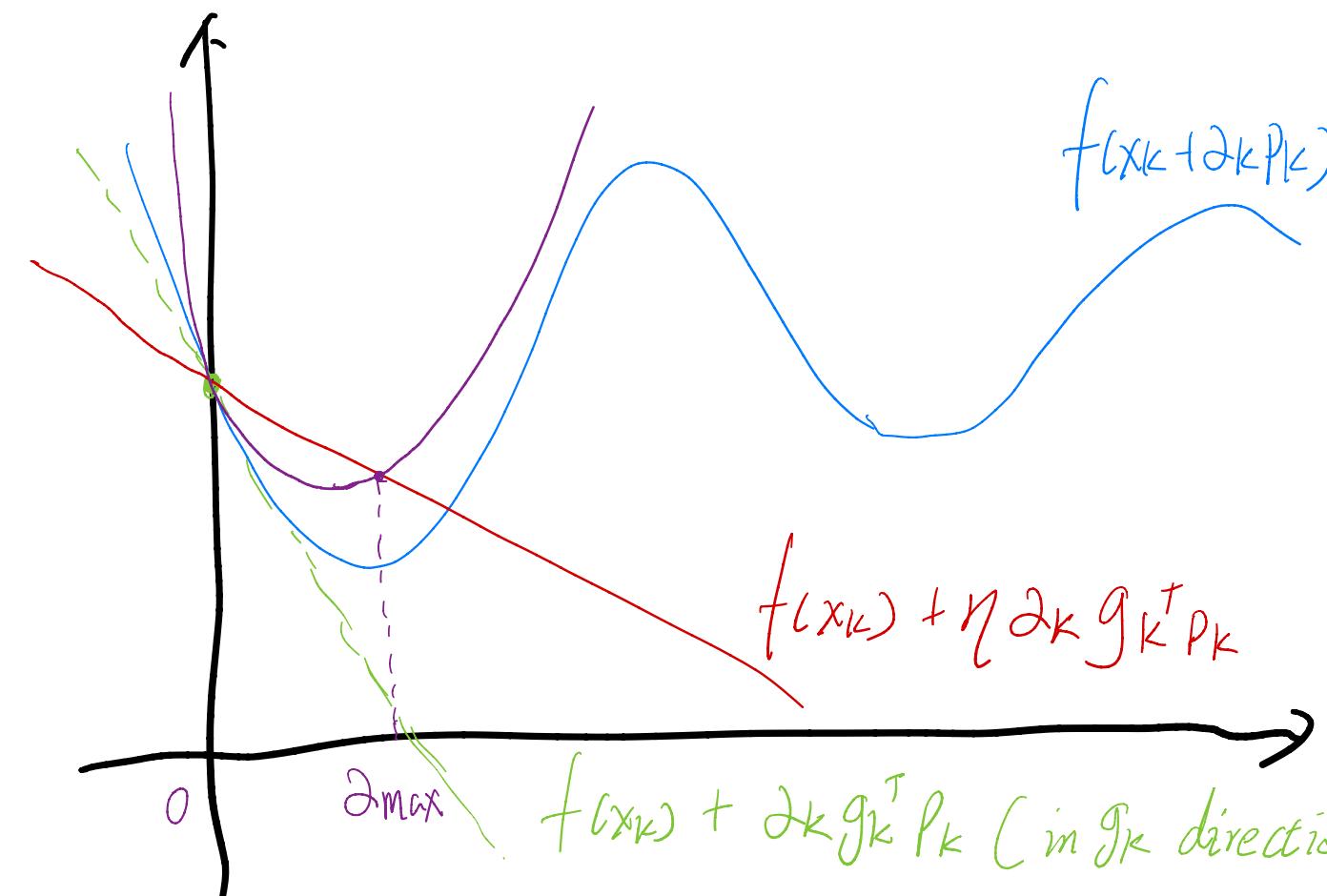
when $\eta = 0 \rightarrow f(x_k)$ flat line

when $\eta = 1 \rightarrow$ tangent.

For different η , we have different Armijo interval.

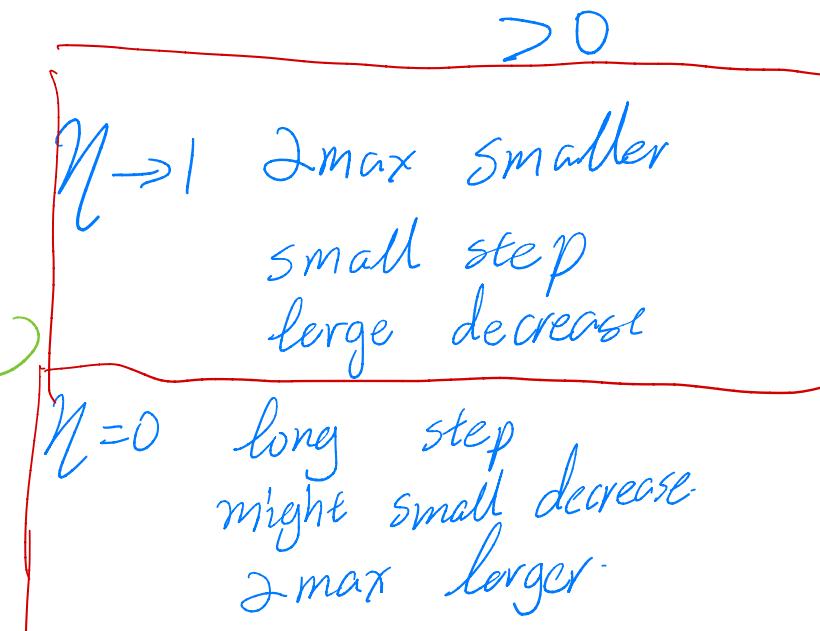
$$f(x_k + \Delta_k p_k) \leq f(x_k) + \eta \Delta_k g_k^T p_k \quad \leftarrow \text{Armijo condition}$$

$$f(x_k + \Delta_k p_k) \leq f(x_k) + \Delta_k g(x_k)^T p_k + \frac{1}{2} \gamma^L(x_k) \|p_k\|_2^2 \rightarrow \text{Taylor}$$



To find Δ_{\max}

$$\Rightarrow \Delta = \Delta_{\max} = \frac{2(\eta - 1)g_k^T p_k}{\gamma^L(x) \|p_k\|_2^2}$$



Theorem 3.2

[Bound on α_k when using an Armijo backtracking linesearch] Suppose that

- $f \in C^1$ and $g(x)$ is Lipschitz continuous with Lipschitz constant γ_k at x_k
- p_k is a descent direction at x_k

Then for any chosen $\eta \in (0, 1)$ the stepsize α_k generated by the backtracking-Armijo linesearch terminates with

$$\alpha_k \geq \min \left(\alpha_{\text{init}}, \frac{2\tau(\eta - 1)g_k^T p_k}{\gamma_k \|p_k\|_2^2} \right) \equiv \min (\alpha_{\text{init}}, \tau \alpha_{\max}(x_k))$$

where $\tau \in (0, 1)$ is the backtracking parameter.

Proof:

Theorem 3.1 implies that the backtracking will terminate as soon as $\alpha^{(l)} \leq \alpha_{\max}(x_k)$. Consider the following two cases:

- α_{init} satisfies the Armijo condition: it is then clear that $\alpha_k = \alpha_{\text{init}}$.
- α_{init} does not satisfy the Armijo condition: thus, there must be a last linesearch iterate (the l -th) for which

$$\alpha^{(l)} > \alpha_{\max}(x_k) \implies \alpha_k = \alpha^{(l+1)} = \tau \alpha^{(l)} > \tau \alpha_{\max}(x_k)$$

Combining the two cases gives the required result. ■

62/106

Theorem 3.3 (Global convergence of generic backtracking-Armijo linesearch)

Suppose that

- $f \in C^1$
- g is Lipschitz continuous on \mathbb{R}^n with Lipschitz constant γ

Then, the iterates generated by the generic backtracking-Armijo linesearch method

Algorithm 11 must satisfy one of the following conditions:

- ① finite termination, i.e., there exists a positive integer k such that

$$g_k = 0 \text{ for some } k \geq 0$$

- ② objective is unbounded below, i.e.,

$$\lim_{k \rightarrow \infty} f_k = -\infty$$

- ③ an angle condition between g_k and p_k exists, i.e.,

$$\lim_{k \rightarrow \infty} \min \left(|p_k^T g_k|, |p_k^T g_k|^2 / \|p_k\|_2^2 \right) = 0$$

$p_k \perp g_k$

$\|g_k\|$ is small
close to 0

64/106

Algorithm 11 Generic backtracking-Armijo linesearch method

-
- 1: **input** initial guess x_0
 - 2: Set $k = 0$
 - 3: **loop**
 - 4: Find a descent direction p_k at x_k .
 - 5: Compute the stepsize α_k using the backtracking-Armijo linesearch Algorithm 10.
 - 6: Set $x_{k+1} = x_k + \alpha_k p_k$
 - 7: Set $k \leftarrow k + 1$
 - 8: **end loop**
 - 9: **return** approximate solution x_k

- should include a sensible relative stopping tolerance such as

$$\|\nabla f(x_k)\| \leq 10^{-8} \max (1, \|\nabla f(x_0)\|)$$

- should also terminate if a maximum number of allowed iterations is reached
- should also terminate if a maximum time limit is reached

we can prove convergence.
and a relates #

Proof:

Suppose that outcomes 1 and 2 do not happen, i.e., $g_k \neq 0$ for all $k \geq 0$ and

$$\lim_{k \rightarrow \infty} f_k > -\infty. \quad (8)$$

The Armijo condition implies that

$$f_{k+1} - f_k \leq \eta \alpha_k p_k^T g_k \approx C |p_k^T g_k|$$

for all k . Summing over the first j iterations yields

$$f_{j+1} - f_0 \leq \eta \sum_{k=0}^j \alpha_k p_k^T g_k \quad (9)$$

Taking limits of the previous inequality as $j \rightarrow \infty$ and using (8) implies that the LHS is bounded below and, therefore, the RHS is also bounded below. Since the sum is composed of all negative terms, we deduce that the following sum

$$\sum_{k=0}^{\infty} \alpha_k |p_k^T g_k| \quad (10)$$

is bounded.

$2k$ is low about

65/106

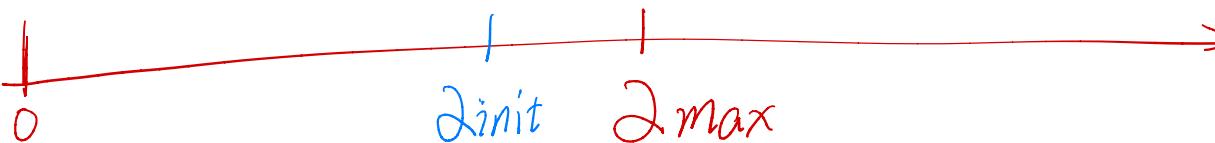
Theorem 3.2 (we don't know λ_{\max} usually, because $\gamma^L(x)$ is unknown)

If λ_{init} lies within $[0, \lambda_{\max}]$

λ_{init} must satisfy

\Rightarrow Armijo condition, then

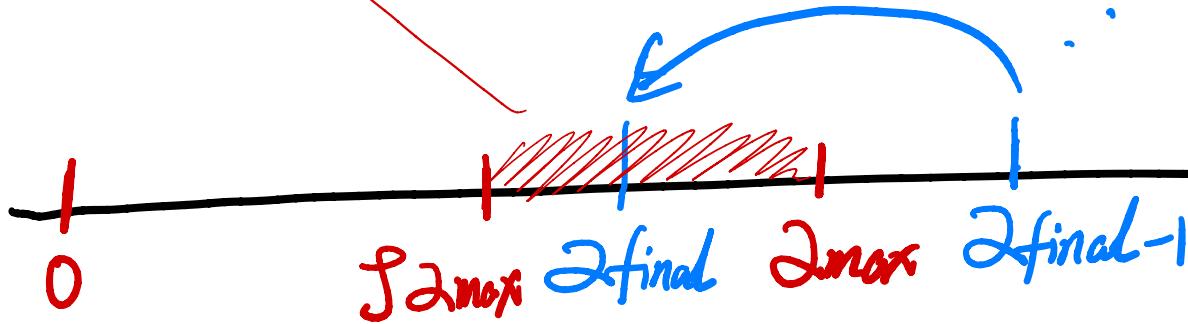
Backtracking algorithm
is terminated.



If λ_{init} lies outside $[0, \lambda_{\max}]$

λ_{final} must lies within

$$[\lambda_{\max}, \lambda_{\max}]$$



If λ_{init} doesn't satisfy
Armijo condition, start

↑ Backtracking



could satisfy Armijo
outside $[0, \lambda_{\max}]$

$[0, 2\max]$ Guarantee Armijo Condition
↓

BUT NOT THE ONLY CASE

There could be some intervals outside $[0, 2\max]$ which also satisfy Armijo condition and make a better step to find the solution.

$\int \in [0, 1]$
 \int value is a trade-off

$\left. \begin{array}{l} \text{close to 0} \rightarrow \text{step might be too small} \\ \text{fast process.} \\ \text{close to 1} \quad \rightarrow \text{too many iterates} \rightarrow \text{expensive} \end{array} \right\}$

From Theorem 3.2, $\alpha_k \geq \min \left\{ \alpha_{init}, \frac{2\tau(\eta-1)g_k^T p_k}{\gamma \|p_k\|_2^2} \right\}$. Thus,

$$\begin{aligned}\sum_{k=0}^{\infty} \alpha_k |p_k^T g_k| &\geq \sum_{k=0}^{\infty} \min \left\{ \alpha_{init}, \frac{2\tau(\eta-1)g_k^T p_k}{\gamma \|p_k\|_2^2} \right\} |p_k^T g_k| \\ &= \sum_{k=0}^{\infty} \min \left\{ \alpha_{init} |p_k^T g_k|, \frac{2\tau(1-\eta) |p_k^T g_k|^2}{\gamma \|p_k\|_2^2} \right\} \\ &\geq \sum_{k=0}^{\infty} \min \left\{ \alpha_{init}, \frac{2\tau(1-\eta)}{\gamma} \right\} \min \left\{ |p_k^T g_k|, \frac{|p_k^T g_k|^2}{\|p_k\|_2^2} \right\}\end{aligned}$$

↗ steepest descent
↗ $p_k = -g_k$
 $\min(|g_k|^2, \|g_k\|^2) = 0$

Using the fact that the series (10) converges, we obtain that

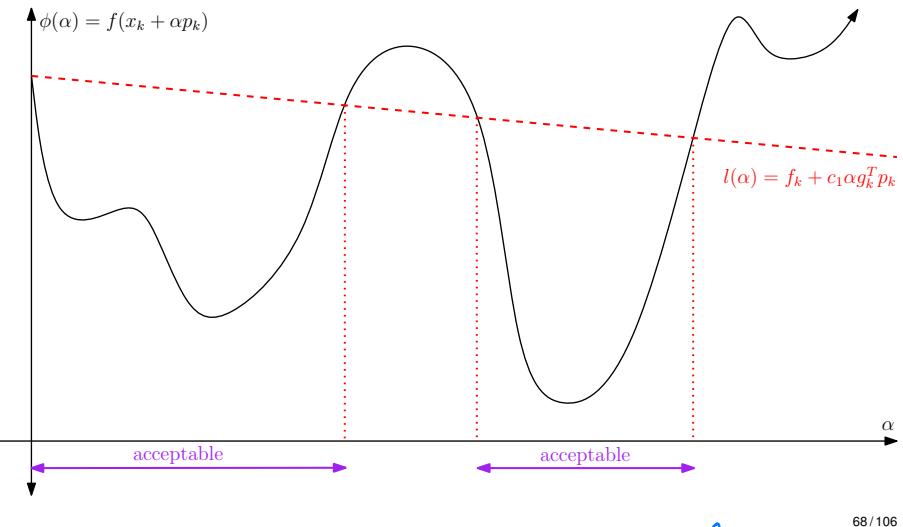
$$\lim_{k \rightarrow \infty} \min \left(|p_k^T g_k|, |p_k^T g_k|^2 / \|p_k\|_2^2 \right) = 0$$

↳ more iterates
↓
we must find $\|g_k\| = 0$

66/106

Some observations

- the Armijo condition prevents too “long” of steps
- typically, the Armijo condition is satisfied for very “large” intervals
- the Armijo condition can be satisfied by steps that are not even close to a minimizer of $\phi(x) := f(x_k + \alpha p_k)$



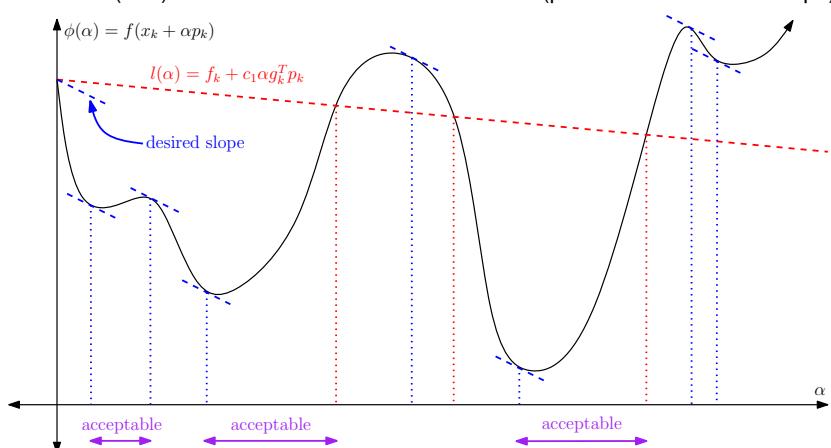
Wolfe conditions

Given the current iterate x_k , search direction p_k , and constants $0 < c_1 < c_2 < 1$, we say that the step length α_k satisfies the **Wolfe conditions** if

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f(x_k)^T p_k \quad (11a)$$

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f(x_k)^T p_k \quad (11b)$$

- condition (11a) is the **Armijo condition** (ensures “sufficient” decrease)
- condition (11b) is a **directional derivative condition** (prevents too short of steps)



Algorithm 12 Generic Wolfe linesearch method

```

1: input initial guess  $x_0$ 
2: Set  $k = 0$ 
3: loop
4:   Find a descent direction  $p_k$  at  $x_k$ .
5:   Compute a stepsize  $\alpha_k$  that satisfies the Wolfe conditions (11).
6:   Set  $x_{k+1} = x_k + \alpha_k p_k$ 
7:   Set  $k \leftarrow k + 1$ 
8: end loop
9: return approximate solution  $x_k$ 

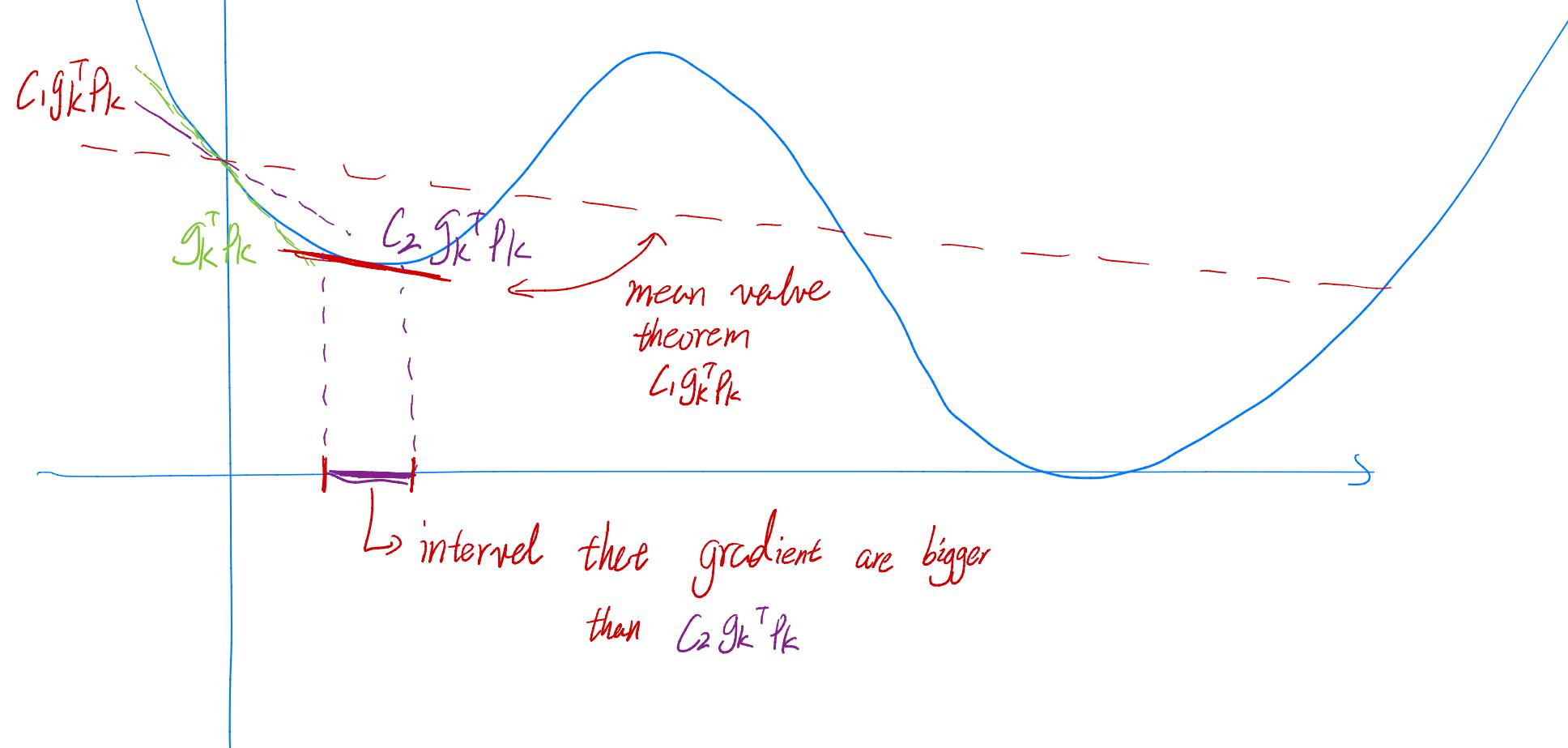
```

- should include a sensible relative stopping tolerance such as

$$\|\nabla f(x_k)\| \leq 10^{-8} \max(1, \|\nabla f(x_0)\|)$$

- should also terminate if a maximum number of allowed iterations is reached
- should also terminate if a maximum time limit is reached

$$0 < C_1 < C_2 < 1$$



Theorem 3.4 (Convergence of generic Wolfe linesearch (Zoutendijk))

Assume that

- $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is bounded below on C^1 on \mathbb{R}^n
- $\nabla f(x)$ is Lipschitz continuous on \mathbb{R}^n with constant L

It follows that the iterates generated from the generic Wolfe linesearch method satisfy

$$\sum_{k \geq 0} \cos^2(\theta_k) \|g_k\|_2^2 < \infty \quad (12)$$

where

$$\cos(\theta_k) \stackrel{\text{def}}{=} \frac{-g_k^T p_k}{\|g_k\|_2 \|p_k\|_2} \quad \text{analogy to } \lim_{k \rightarrow \infty} (\theta_k) = 0 \quad (13)$$

Comments

- definition (13) measures the angle between the gradient g_k and search direction p_k
 - $\cos(\theta) \approx 0$ implies g_k and p_k are nearly orthogonal
 - $\cos(\theta) \approx 1$ implies g_k and p_k are nearly parallel
- condition (12) ensures that the gradient converges to zero provided g_k and p_k stay away from being arbitrarily close to orthogonal

Proof: See Theorem 3.2 in Nocedal and Wright.

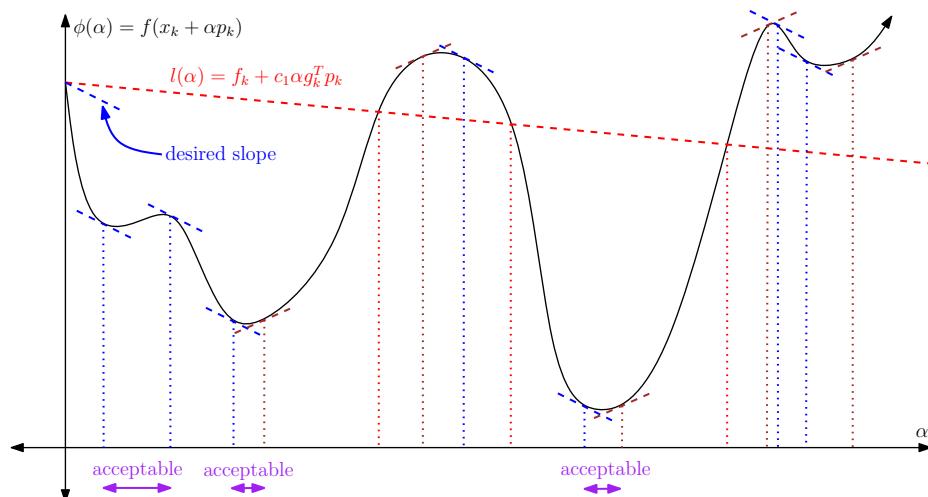
71 / 106

Strong Wolfe conditions

Given the current iterate x_k , search direction p_k , and constants $0 < c_1 < c_2 < 1$, we say that the step length α_k satisfies the **Strong Wolfe conditions** if

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f(x_k)^T p_k \quad (14a)$$

$$|\nabla f(x_k + \alpha_k p_k)^T p_k| \leq c_2 |\nabla f(x_k)^T p_k| \quad (14b)$$



74 / 106

Some comments

- The Armijo linesearch
 - only requires function evaluations
 - may easily accept steps that are far from a univariate minimizer of $\phi(\alpha) = f(x_k + \alpha p_k)$
- The Wolfe linesearch
 - requires function and gradient evaluations
 - typically produces steps that are closer to the univariate minimizer of ϕ
 - may still allow the acceptance of steps that are far from a univariate minimizer of ϕ
 - "ideal" when search directions are computed from linear systems based on the BFGS quasi-Newton updating formula

Note: we have not yet said how to compute steps that satisfy the Wolfe conditions. We have not even shown that such steps exist! (coming soon)

72 / 106

Theorem 3.5

Suppose that $f \in \mathbb{R}^n \rightarrow \mathbb{R}$ is in C^1 and p_k is a **descent direction** at x_k , and assume that f is bounded below along the ray $\{x_k + \alpha p_k : \alpha > 0\}$. It follows that if $0 < c_1 < c_2 < 1$, then there exist nontrivial intervals of step lengths that satisfy the **strong Wolfe conditions** (14).

Proof:

The function $\phi(\alpha) := f(x_k + \alpha p_k)$ is bounded below over $\alpha > 0$ by assumption and must, therefore, intersect the graph of $l(\alpha) := f(x_k) + \alpha c_1 \nabla f(x_k)^T p_k$ at least once since $c_1 \in (0, 1)$; let $\alpha_{\min} > 0$ be the smallest such intersecting value so that

$$f(x_k + \alpha_{\min} p_k) = f(x_k) + \alpha_{\min} c_1 \nabla f(x_k)^T p_k. \quad (15)$$

It is clear that (11a)/(14a) hold for all $\alpha \in (0, \alpha_{\min}]$. The Mean Value Theorem now implies

$$f(x_k + \alpha_{\min} p_k) - f(x_k) = \alpha_{\min} \nabla f(x_k + \alpha_M p_k)^T p_k \quad \text{for some } \alpha_M \in (0, \alpha_{\min}).$$

Combining the last equality with (15) and the facts that $c_1 < c_2$ and $\nabla f(x_k)^T p_k < 0$ yield

$$\nabla f(x_k + \alpha_M p_k)^T p_k = c_1 \nabla f(x_k)^T p_k > c_2 \nabla f(x_k)^T p_k. \quad (16)$$

Thus, α_M satisfies (11a)/(14a) and (11b) with a **strict** inequality so we may use the smoothness assumption to deduce that there is an interval around α_M for which (11) and (14a) hold. Finally, since the left-hand-side of (16) is negative, we also know that (14b) holds, which completes the proof. ■

75 / 106

Algorithm 13 A bisection type (weak) Wolfe linesearch

```

1: input  $x_k, p_k, 0 < c_1 < c_2 < 1$ 
2: Choose  $\alpha = 0, t = 1$ , and  $\beta = \infty$ 
3: while  $(f(x_k + tp_k) > f(x_k) + c_1 \cdot t \cdot (g_k^T p_k)$  OR  $\nabla f(x_k + tp_k)^T p_k < c_2(g_k^T p_k)$  do
4:   if  $f(x_k + tp_k) > f(x_k) + c_1 \cdot t \cdot (g_k^T p_k)$  then
5:     Set  $\beta := t$ 
6:     Set  $t := \frac{\alpha+\beta}{2}$ 
7:   else
8:      $\alpha := t$ 
9:     if  $\beta = \infty$  then
10:       $t := 2\alpha$ 
11:    else
12:       $t := \frac{\alpha+\beta}{2}$ 
13:    end if
14:   end if
15: end while
16:
17: return  $t$ 

```

mean value theorem

$$\frac{f(b) - f(a)}{b-a} = f'(s)$$

at a, b

See Section 3.5 in Nocedal-Wright for more discussions; in particular, a procedure for the **strong** Wolfe conditions.

76 / 106

Theorem 3.6 (Wolfe conditions guarantee the BFGS update is well-defined)

Suppose that $B_k \succ 0$ and that p_k is a descent direction for f at x_k . If a (strong) Wolfe linesearch is used to compute α_k such that $x_{k+1} = x_k + \alpha_k p_k$, then

$$y_k^T s_k > 0$$

where

$$s_k = x_{k+1} - x_k = \alpha_k p_k \quad \text{and} \quad y_k = g_{k+1} - g_k.$$

Proof:

It follows from (11b) that

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f(x_k)^T p_k.$$

Multiplying both sides by α_k , using $x_{k+1} = x_k + \alpha_k p_k$, and notation $g_k = \nabla f(x_k)$, yields

$$g_{k+1}^T s_k \geq c_2 g_k^T s_k.$$

Subtracting $g_k^T s_k$ from both sides gives

$$g_{k+1}^T s_k - g_k^T s_k \geq c_2 g_k^T s_k - g_k^T s_k$$

so that

$$y_k^T s_k \geq (c_2 - 1) g_k^T s_k = \alpha_k (c_2 - 1) g_k^T p_k > 0$$

since $c_2 \in (0, 1)$ and p_k is a descent direction for f at x_k . ■

Some comments

- The Armijo linesearch
 - ▶ only requires function evaluations
 - ▶ computationally cheapest to implement
 - ▶ may easily accept steps that are far from a univariate minimizer of $\phi(\alpha) = f(x_k + \alpha p_k)$
- The Wolfe linesearch
 - ▶ requires function **and** gradient evaluations
 - ▶ typically produces steps that are closer to the univariate minimizer of ϕ
 - ▶ may still allow the acceptance of steps that are far from a univariate minimizer of ϕ
 - ▶ computationally more expensive to implement compared to Armijo
- The strong Wolfe linesearch
 - ▶ requires function **and** gradient evaluations
 - ▶ typically produces steps that are closer to the univariate minimizer of ϕ
 - ▶ only approximate stationary points of the univariate function ϕ satisfy these conditions.
 - ▶ computationally most expensive to implement compared to Armijo

Why use (strong) Wolfe conditions?

- sometimes more accurate line search lead to fewer outer (major) iterations of the linesearch method.
- beneficial in the context of quasi-Newton methods for maintaining positive-definite matrix approximations.

77 / 106

Algorithm 14 Steepest descent backtracking-Armijo linesearch method

```

1: Input: initial guess  $x_0$ 
2: Set  $k = 0$ 
3: until  $\|\nabla f(x_k)\| \leq 10^{-8} \max(1, \|\nabla f(x_0)\|)$ 
4:   Set  $p_k = -\nabla f(x_k)$  as the search direction.
5:   Compute the stepsize  $\alpha_k$  using the backtracking-Armijo linesearch Algorithm 10.
6:   Set  $x_{k+1} = x_k + \alpha_k p_k$ 
7:   Set  $k \leftarrow k + 1$ 
8: end until
9: return approximate solution  $x_k$ 

```

- should also terminate if a maximum number of allowed iterations is reached
- should also terminate if a maximum time limit is reached

Theorem 4.1 (Global convergence for steepest descent)

If $f \in C^1$ and \mathbf{g} is Lipschitz continuous on \mathbb{R}^n , then the iterates generated by the steepest-descent backtracking-Armijo linesearch Algorithm 14 satisfies one of the following conditions:

- ① **finite termination**, i.e., there exists some finite integer $k \geq 0$ such that

$$\mathbf{g}_k = \mathbf{0}$$

- ② **objective is unbounded below**, i.e.,

$$\lim_{k \rightarrow \infty} f_k = -\infty$$

- ③ **convergence of gradients**, i.e.,

$$\lim_{k \rightarrow \infty} \mathbf{g}_k = \mathbf{0}$$

Proof:

Since $p_k = -g_k$, if outcome 1 above does not happen, then we may conclude that

$$\|\mathbf{g}_k\| = \|p_k\| \neq 0 \text{ for all } k \geq 0$$

Additionally, if outcome 2 above does not occur, then Theorem 3.3 ensures that

$$0 = \lim_{k \rightarrow \infty} \left\{ \min \left(|p_k^T g_k|, |p_k^T g_k|^2 / \|p_k\|_2^2 \right) \right\} = \lim_{k \rightarrow \infty} \{ \|\mathbf{g}_k\|_2 \min (\|\mathbf{g}_k\|_2, 1) \}$$

and thus $\lim_{k \rightarrow \infty} \mathbf{g}_k = \mathbf{0}$. ■

82/106

Rate of convergence

Theorem 4.2 (Global convergence rate for steepest-descent)

Suppose $f \in C^1$ and ∇f is Lipschitz continuous on \mathbb{R}^n . Further assume that $f(x)$ is bounded from below on \mathbb{R}^n . Let $\{x_k\}$ be the iterates generated by the steepest-descent backtracking-Armijo linesearch Algorithm 14. Then there exists a constant M such that for all $T \geq 1$

$$\min_{k=0, \dots, T} \|\nabla f(x_k)\| \leq \frac{M}{\sqrt{T+1}}.$$

Consequently, for any $\epsilon > 0$, within $\lceil (\frac{M}{\epsilon})^2 \rceil$ steps, we will see an iterate where the gradient has norm at most ϵ . In other words, we reach an " ϵ -stationary" point in $O((\frac{1}{\epsilon})^2)$ steps

REMARK: Same thing holds for constant step size for an appropriately chosen constant. See HW.

Proof:

Recall (9) in the proof of Theorem 3.3:

$$\sum_{k=0}^T \alpha_k |\mathbf{g}_k^T p_k| \leq \frac{f_0 - f_{k+1}}{\eta} \leq M' \quad \text{since } f \text{ is bounded from below.}$$

84/106

Some general comments on the steepest descent backtracking-Armijo algorithm

- archetypal globally convergent method
- many other methods resort to steepest descent when things "go wrong"
- not scale invariant
- convergence is usually slow! (linear local convergence, possibly with constant close to 1; for global convergence only sublinear rate $O((\frac{1}{\epsilon})^2)$ to get to a "stationary" point, i.e., $\|\mathbf{g}_k\| \leq \epsilon$)
- in practice, often does not converge at all
- slow convergence results because the computation of p_k does not use any curvature information
- call any method that uses the steepest descent direction a "method of steepest descent"
- each iteration is relatively cheap, but you may need many of them!

83/106

Rate of convergence

Thus,

$$\sum_{k: \tau \alpha_{\max} < \alpha_{\text{init}}} \frac{2\tau(1-\eta)|\mathbf{g}_k^T p_k|^2}{\gamma \|p_k\|^2} + \sum_{k: \tau \alpha_{\max} \geq \alpha_{\text{init}}} \alpha_{\text{init}} |\mathbf{g}_k^T p_k| \leq M'.$$

Thus,

$$\sum_{k: \tau \alpha_{\max} < \alpha_{\text{init}}} \frac{|\mathbf{g}_k^T p_k|^2}{\|p_k\|^2} + \sum_{k: \tau \alpha_{\max} \geq \alpha_{\text{init}}} |\mathbf{g}_k^T p_k| \leq \frac{M'}{C}, \quad (17)$$

$$\text{where } C = \min \left\{ \frac{2\tau(1-\eta)}{\gamma}, \alpha_{\text{init}} \right\}.$$

Setting $p_k = -g_k$ in the inequality above, we obtain that

$$(T+1) \min_{k=0, \dots, T} \|\mathbf{g}_k\|^2 \leq \sum_{k=0, \dots, T} \|\mathbf{g}_k\|^2 \leq \frac{M'}{C}.$$

Therefore,

$$\min_{k=0, \dots, T} \|\mathbf{g}_k\| \leq \frac{M}{\sqrt{T+1}},$$

for some constant M . ■

85/106

Theorem 4.2 Global convergence rate for steepest descent.

$$\min_{x \in \mathbb{R}^n} f(x)$$

Assume $\exists B \leq \text{fix} \quad \forall x \in \mathbb{R}^n$

$$x_0, x_1, \dots, x_k, \dots, x_{k+1}$$

$$x_{k+1} = x_k + \alpha_k p_k$$

Suppose: $\exists C > 0 \quad f(x_{k+1}) \leq f(x_k) - C \|g_k\|^2 \quad \forall k=0, 1, \dots$

sufficient decrease.

the algorithm generate $T+1$ iterates: $x_0, x_1, \dots, x_k, \dots, x_{T-1}, \dots, x_T$

For each iterates

$$\left. \begin{aligned} f(x_T) &\leq f(x_{T-1}) - C \|g_{T-1}\|^2 \\ f(x_{T-1}) &\leq f(x_{T-2}) - C \|g_{T-2}\|^2 \\ &\vdots \\ f(x_1) &\leq f(x_0) - C \|g_0\|^2 \end{aligned} \right\} \sum_{i=0}^{k-1} B \leq \Rightarrow f(x_T) - f(x_0) \leq - \sum_{i=0}^{T-1} C \|g_i\|^2$$

$$\Rightarrow f(x_0) - B \geq \sum_{i=0}^{T-1} C \|g_i\|^2$$

B is global low bound. $f(x) \geq B$

$$\Rightarrow \cancel{\sum_{i=0}^{T-1} \|g_k\|^2} \leq \boxed{\frac{f(x_0) - \beta}{C}} = M$$

$$\Rightarrow T \min_{k=0, \dots, T-1} \|g_k\|^2 \leq M$$

$$\Rightarrow \min_{k=0, \dots, T-1} (\|g_k\|^2) \leq \frac{M}{T}$$

as $T \rightarrow \infty \Rightarrow \min \|g_k\|^2 \rightarrow 0$
function is bounded

Steepest descent example

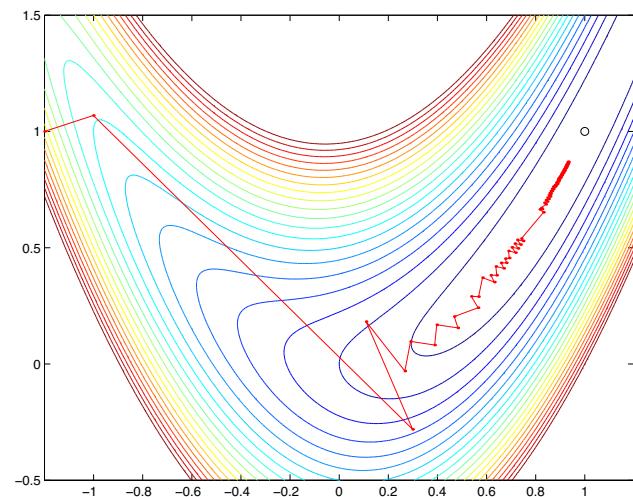


Figure: Contours for the objective function $f(x, y) = 10(y - x^2)^2 + (x - 1)^2$, and the iterates generated by the steepest-descent backtracking-Armijo linesearch Algorithm 14.

86 / 106

Algorithm 15 Modified-Newton backtracking-Armijo linesearch method

```

1: Input: initial guess  $x_0$ 
2: Set  $k = 0$ 
3: until  $\|\nabla f(x_k)\| \leq 10^{-8} \max(1, \|\nabla f(x_0)\|)$ 
4: Compute positive-definite matrix  $B_k$  from  $H_k$  using Algorithm 2.
5: Compute the search direction  $p_k$  as the solution to  $B_k p = -g_k$ .
6: Compute the stepsize  $\alpha_k$  using the backtracking-Armijo linesearch Algorithm 10.
7: Set  $x_{k+1} = x_k + \alpha_k p_k$ 
8: Set  $k \leftarrow k + 1$ 
9: end until
10: return approximate solution  $x_k$ 
```

- should also terminate if a maximum number of allowed iterations is reached
- should also terminate if a maximum time limit is reached

89 / 106

Rate of convergence

Theorem 4.3 (Local convergence rate for steepest-descent)

Suppose the following all hold:

- f is twice continuously differentiable and $\nabla^2 f$ is Lipschitz continuous on \mathbb{R}^n with Lipschitz constant M .
- x^* is a local minimum of f with positive definite Hessian $\nabla^2 f(x^*)$ with eigenvalues lower bounded by ℓ and upper bounded by L .
- The initial starting iterate x_0 is close enough to x^* ; more precisely,
 $r_0 := \|x_0 - x^*\| < \bar{r} := \frac{2\ell}{M}$.

The steepest-descent with constant step size $\alpha = \frac{2}{\ell+L}$ converges as follows:

$$\|x_k - x^*\| \leq \frac{\bar{r}r_0}{\bar{r} - r_0} \left(1 - \frac{2\ell}{L+3\ell}\right)^k.$$

In other words, in $O(\log(\frac{1}{\epsilon}))$ iterations, we are within a distance of ϵ from x^* .

Proof: See Theorem 1.2.4 in Nesterov's book.

See also Theorem 3.4 in Nocedal-Wright for a similar result with exact line-search.

87 / 106

Theorem 4.4 (Global convergence of modified-Newton backtracking-Armijo)

Suppose that

- $f \in C^1$
- g is Lipschitz continuous on \mathbb{R}^n
- the eigenvalues of B_k are uniformly bounded away from zero and infinity

Then, the iterates generated by the modified-Newton backtracking-Armijo Algorithm 15 satisfy one of the following:

- ① **finite termination**, i.e., there exists a finite k such that

$$g_k = 0$$

- ② **objective is unbounded below**, i.e.,

$$\lim_{k \rightarrow \infty} f_k = -\infty$$

- ③ **convergence of gradients**, i.e.,

$$\lim_{k \rightarrow \infty} g_k = 0$$

can be shown

Proof:

Let $\lambda_{\min}(B_k)$ and $\lambda_{\max}(B_k)$ be the smallest and largest eigenvalues of B_k . By assumption, there are bounds $0 < \lambda_{\min} \leq \lambda_{\max}$ such that

$$\lambda_{\min} \leq \lambda_{\min}(B_k) \leq \frac{s^T B_k s}{\|s\|^2} \leq \lambda_{\max}(B_k) \leq \lambda_{\max} \quad \text{for any nonzero } s.$$

90 / 106

From the previous slide, it follows that

$$\lambda_{\max}^{-1} \leq \lambda_{\max}^{-1}(B_k) = \lambda_{\min}(B_k^{-1}) \leq \frac{s^T B_k^{-1} s}{\|s\|^2} \leq \lambda_{\max}(B_k^{-1}) = \lambda_{\min}^{-1}(B_k) \leq \lambda_{\min}^{-1} \quad (18)$$

for any nonzero vector s . Using $B_k p_k = -g_k$ and (18) we see that

$$|p_k^T g_k| = |g_k^T B_k^{-1} g_k| \geq \lambda_{\max}^{-1} \|g_k\|_2^2 \quad (19)$$

In addition, we may use (18) to see that

$$\|p_k\|_2^2 = g_k^T B_k^{-2} g_k \leq \lambda_{\min}^{-2} \|g_k\|_2^2,$$

which implies that

$$\|p_k\|_2 \leq \lambda_{\min}^{-1} \|g_k\|_2.$$

It now follows from the previous inequality and (19) that

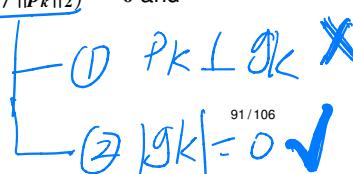
$$\frac{|p_k^T g_k|}{\|p_k\|_2} \geq \frac{\lambda_{\min}}{\lambda_{\max}} \|g_k\|_2. \quad (20)$$

Combining the previous inequality and (19) yields

$$\min(|p_k^T g_k|, |p_k^T g_k|^2 / \|p_k\|_2^2) \geq \frac{\|g_k\|_2^2}{\lambda_{\max}} \min\left(1, \frac{\lambda_{\min}^2}{\lambda_{\max}}\right). \quad (21)$$

From Theorem 3.3 we know that $\lim_{k \rightarrow \infty} \min(|p_k^T g_k|, |p_k^T g_k|^2 / \|p_k\|_2^2) = 0$ and combined with (21), we conclude that

$$\lim_{k \rightarrow \infty} g_k = 0. \quad \blacksquare$$



Rate of convergence

Theorem 4.6 (Local quadratic convergence for modified-Newton)

Suppose that

- $f \in C^2$
 - H is Lipschitz continuous on \mathbb{R}^n with constant γ
- Suppose that iterates are generated from the modified-Newton backtracking-Armijo Algorithm 15 such that
- $\alpha_{\text{init}} = 1$
 - $\eta \in (0, \frac{1}{2})$
 - $B_k = H_k$ whenever H_k is positive definite

If the sequence of iterates $\{x_k\}$ has a limit point x^* satisfying $H(x^*)$ is positive definite, then it follows that

- (i) $\alpha_k = 1$ for all sufficiently large k ,
- (ii) the entire sequence $\{x_k\}$ converges to x^* , and
- (iii) the rate of convergence is q -quadratic, i.e. there is a constant $\kappa \geq 0$ and a natural number k_0 such that for all $k \geq k_0$

$$\|x_{k+1} - x^*\|_2 \leq \kappa \|x_k - x^*\|_2^2$$

Rate of convergence

Theorem 4.5 (Global convergence rate for modified-Newton) *upper bound*

Suppose $f \in C^1$ and ∇f is Lipschitz continuous on \mathbb{R}^n . Further assume that $f(x)$ is bounded from below on \mathbb{R}^n . Let $\{x_k\}$ be the iterates generated by the modified-Newton backtracking-Armijo linesearch Algorithm 15 such that B_k has eigenvalues uniformly bounded away from 0 and infinity. Then there exists a constant M such that for all $T \geq 1$

$$\min_{k=0, \dots, T} \|\nabla f(x_k)\| \leq \frac{M}{\sqrt{T+1}}.$$

Consequently, for any $\epsilon > 0$, within $\lceil (\frac{M}{\epsilon})^2 \rceil$ steps, we will see an iterate where the gradient has norm at most ϵ . In other words, we reach an " ϵ -stationary" point in $O((\frac{1}{\epsilon})^2)$ steps

Proof: Almost identical to steepest descent proof (Theorem 4.2); use (19) and (20) in (17).

REMARK: Same thing holds for constant step size for an appropriately chosen constant.

92/106

Proof:

By assumption we know that there is some subsequence \mathcal{K} such that

$$\lim_{k \in \mathcal{K}} x_k = x^* \quad \text{and} \quad H(x^*) \succ 0$$

Continuity implies that H_k is positive definite for all $k \in \mathcal{K}$ sufficiently large so that

$$p_k^T H_k p_k \geq \frac{1}{2} \lambda_{\min}(H(x^*)) \|p_k\|_2^2 \quad \text{for all } k_0 \leq k \in \mathcal{K} \quad (22)$$

for some k_0 , where $\lambda_{\min}(H(x^*))$ is the smallest eigenvalue of $H(x^*)$. This fact and since $H_k p_k = -g_k$ for $k_0 \leq k \in \mathcal{K}$ implies that

$$|p_k^T g_k| = -p_k^T g_k = p_k^T H_k p_k \geq \frac{1}{2} \lambda_{\min}(H(x^*)) \|p_k\|_2^2 \quad \text{for all } k_0 \leq k \in \mathcal{K}. \quad (23)$$

This also implies

$$\frac{|p_k^T g_k|}{\|p_k\|_2} \geq \frac{1}{2} \lambda_{\min}(H(x^*)) \|p_k\|_2. \quad (24)$$

We now deduce from Theorem 3.3, (23), and (24) that

$$\lim_{k \in \mathcal{K} \rightarrow \infty} p_k = 0.$$

94/106

Mean Value Theorem implies that there exists z_k between x_k and $x_k + p_k$ such that

$$f(x_k + p_k) = f_k + p_k^T g_k + \frac{1}{2} p_k^T H(z_k) p_k.$$

Lipschitz continuity of $H(x)$ and the fact that $H_k p_k = -g_k$ implies

$$\begin{aligned} f(x_k + p_k) - f_k - \frac{1}{2} p_k^T g_k &= \frac{1}{2} (p_k^T g_k + p_k^T H(z_k) p_k) \\ &= \frac{1}{2} p_k^T (H(z_k) - H_k) p_k \\ &\leq \frac{1}{2} \|H(z_k) - H(x_k)\|_2 \|p_k\|_2^2 \leq \frac{1}{2} \gamma \|p_k\|_2^3. \end{aligned} \quad (25)$$

Since $\eta \in (0, 1/2)$ and $p_k \rightarrow 0$, we may pick k sufficiently large so that

$$2\gamma \|p_k\|_2 \leq \lambda_{\min}(H(x^*)) (1 - 2\eta). \quad (26)$$

Combining (25), (26), and (23) shows that

$$\begin{aligned} f(x_k + p_k) - f_k &\leq \frac{1}{2} p_k^T g_k + \frac{1}{2} \gamma \|p_k\|_2^3 \\ &\leq \frac{1}{2} p_k^T g_k + \frac{1}{4} \lambda_{\min}(H(x^*)) (1 - 2\eta) \|p_k\|_2^2 \\ &\leq \frac{1}{2} p_k^T g_k - \frac{1}{2} (1 - 2\eta) p_k^T g_k \\ &= \eta p_k^T g_k \end{aligned}$$

so that the unit step $x_k + p_k$ satisfies the Armijo condition for all $k \in \mathcal{K}$ sufficiently large.

95 / 106

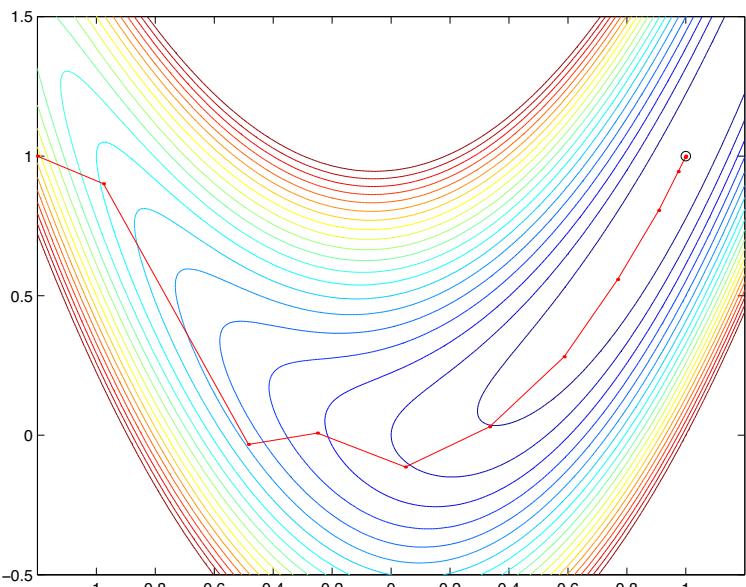


Figure: Contours for the objective function $f(x, y) = 10(y - x^2)^2 + (x - 1)^2$, and the iterates generated by the modified-Newton backtracking-Armijo Algorithm 15.

Note that (22) implies that

$$\|H_k^{-1}\|_2 \leq 2/\lambda_{\min}(H(x^*)) \quad \text{for all } k_0 \leq k \in \mathcal{K}. \quad (27)$$

The iteration gives

$$\begin{aligned} x_{k+1} - x^* &= x_k - x^* - H_k^{-1} g_k = x_k - x^* - H_k^{-1} (g_k - g(x^*)) \\ &= H_k^{-1} [g(x^*) - g_k - H_k(x^* - x_k)]. \end{aligned}$$

But a Taylor Approximation provides the estimate

$$\|g(x^*) - g_k - H_k(x^* - x_k)\|_2 \leq \frac{\gamma}{2} \|x_k - x^*\|_2^2$$

which implies

$$\|x_{k+1} - x^*\|_2 \leq \frac{\gamma}{2} \|H_k^{-1}\|_2 \|x_k - x^*\|_2^2 \leq \frac{\gamma}{\lambda_{\min}(H(x^*))} \|x_k - x^*\|_2^2 \quad (28)$$

Thus, we know that for $k \in \mathcal{K}$ sufficiently large, the unit step will be accepted and

$$\|x_{k+1} - x^*\|_2 \leq \frac{1}{2} \|x_k - x^*\|_2$$

which proves parts (i) and (ii) since the entire argument may be repeated. Finally, part (iii) then follows from part (ii) and (28) with the choice

$$\kappa = \gamma / \lambda_{\min}(H(x^*))$$

which completes the proof. ■

96 / 106

Algorithm 16 Modified-Newton Wolfe linesearch method

-
- 1: Input: initial guess x_0
 - 2: Set $k = 0$
 - 3: **until** $\|\nabla f(x_k)\| \leq 10^{-8} \max(1, \|\nabla f(x_0)\|)$
 - 4: Compute positive-definite matrix B_k from H_k using Algorithm 2.
 - 5: Compute a search direction p_k as the solution to $B_k p = -g_k$.
 - 6: Compute a stepsize α_k that satisfies the Wolfe conditions (11a) and (11b).
 - 7: Set $x_{k+1} \leftarrow x_k + \alpha_k p_k$
 - 8: Set $k \leftarrow k + 1$
 - 9: **end until**
 - 10: **return** approximate solution x_k
-

- should also terminate if a maximum number of allowed iterations is reached
- should also terminate if a maximum time limit is reached

$$g(x^*) = 0$$

Algorithm 17 Quasi-Newton Wolfe linesearch method

```
1: Input: initial guess  $x_0$ 
2: Set  $k = 0$ 
3: until  $\|\nabla f(x_k)\| \leq 10^{-8} \max(1, \|\nabla f(x_0)\|)$ 
4:   Compute positive-definite matrix  $B_k$  from  $B_{k-1}$  using the BFGS rule, or using Algorithm 3 with a seed matrix  $B_k^0$ .
5:   Compute a search direction  $p_k$  as the solution to  $B_k p = -g_k$ .
6:   Compute a stepsize  $\alpha_k$  that satisfies the Wolfe conditions (11a) and (11b).
7:   Set  $x_{k+1} \leftarrow x_k + \alpha_k p_k$ 
8:   Set  $k \leftarrow k + 1$ 
9: end until
10: return approximate solution  $x_k$ 
```

- should also terminate if a maximum number of allowed iterations is reached
- should also terminate if a maximum time limit is reached

101 / 106

Rate of convergence

Theorem 4.8 (Global convergence rate for modified-Newton/quasi-Newton Wolfe)

Suppose $f \in C^1$ and ∇f is Lipschitz continuous on \mathbb{R}^n . Further assume that $f(x)$ is bounded from below on \mathbb{R}^n . Let $\{x_k\}$ be the iterates generated by the modified/quasi-Newton Wolfe lineasearch Algorithms 16 or 17, such that B_k has uniformly bounded condition number. Then there exists a constant M such that for all $T \geq 1$

$$\min_{k=0, \dots, T} \|\nabla f(x_k)\| \leq \frac{M}{\sqrt{T+1}}.$$

Consequently, for any $\epsilon > 0$, within $\lceil (\frac{M}{\epsilon})^2 \rceil$ steps, we will see an iterate where the gradient has norm at most ϵ . In other words, we reach an “ ϵ -stationary” point in $O((\frac{1}{\epsilon})^2)$ steps

Theorem 4.7 (Global convergence of modifed-Newton/quasi-Newton Wolfe)

Suppose that

- $f \in C^1$
- g is Lipschitz continuous on \mathbb{R}^n
- the condition number of B_k is uniformly bounded by β

Then, the iterates generated by the modified/quasi-Newton Wolfe Algorithm 17 or 16 satisfy one of the following:

- ① finite termination, i.e., there exists a finite k such that

$$g_k = 0$$

- ② objective is unbounded below, i.e.,

$$\lim_{k \rightarrow \infty} f_k = -\infty$$

- ③ convergence of gradients, i.e.,

$$\lim_{k \rightarrow \infty} g_k = 0$$

Proof:

Homework assignment.

102 / 106

Rate of convergence

Theorem 4.9 (Local superlinear convergence for quasi-Newton)

Suppose that

- $f \in C^2$
- H is Lipschitz continuous on \mathbb{R}^n

Suppose that iterates are generated from the quasi-Newton Wolfe Algorithm Algorithm 17 such that $0 < c_1 < c_2 \leq \frac{1}{2}$.

If the sequence of iterates $\{x_k\}$ has a limit point x^* satisfying $H(x^*)$ is positive definite and $g(x^*) = 0$, then it follows that

- (i) the step length $\alpha_k = 1$ is valid for all sufficiently large k ,
- (ii) the entire sequence $\{x_k\}$ converges to x^* superlinearly.

See also Theorems 3.6 and 3.7 from Nocedal-Wright.

103 / 106

104 / 106

The problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x)$$

where the **objective function** $f : \mathbb{R}^n \rightarrow \mathbb{R}$

Solve instead quadratic approximation at current iterate x_k , after choosing **positive definite** B_k , to get a **search direction** that is a **descent direction**:

$$p_k = \underset{p \in \mathbb{R}^n}{\operatorname{argmin}} \quad m_k^0(p) \stackrel{\text{def}}{=} f_k + g_k^T p + \frac{1}{2} p^T B_k p$$

- B_k = identity matrix: **Steepest Descent Direction**.
- B_k obtained from Hessian H_k by adjusting eigenvalues: **Modified-Newton**.
- B_k obtained by updating at every iteration using current and previous gradient information: **Quasi-Newton**.

Decide on step size strategy: **Armijo backtracking line search**, **Wolfe backtracking line search**, constant step size (HW assignment), $O\left(\frac{1}{\sqrt{k}}\right)$ in iteration k ... many others.

Steepest Descent	Modified/Quasi Newton
Requires only function and gradient evaluations	Could need Hessian evaluations as well (with spectral decompositions)
No overhead of solving linear system in each iteration	Solves a linear system $Bp = g$ in each iteration
Global convergence to ϵ -stationary point in $O((\frac{1}{\epsilon})^2)$ steps	Global convergence to ϵ -stationary point in $O((\frac{1}{\epsilon})^2)$ steps
Linear local convergence: converges to ϵ distance from local minimum in $O(\log(\frac{1}{\epsilon}))$ steps	Superlinear (quadratic) local convergence: converges to ϵ distance from local minimum in $o(\log(\frac{1}{\epsilon}))$ ($O(\log \log(\frac{1}{\epsilon}))$ steps).