**The problem**

\[
\text{minimize } f(x) \quad \text{subject to } x \in \mathbb{R}^n
\]

where the objective function \( f : \mathbb{R}^n \to \mathbb{R} \)

- 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 \( k = 1, 2, \ldots \)

**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**

- **Generic Line-search Framework**
  - 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
  - Choosing the step length \( \alpha_k \) (linesearch)
    - Backtracking-Armijo linesearch
    - Wolfe conditions
    - Strong Wolfe conditions
- **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

---

**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.,
  \[
  g_k^T p_k < 0 \quad \text{if} \quad g_k \neq 0
  \]
  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 \( \alpha_k \) is the linesearch
- the computation of \( \alpha_k \) may itself require an iterative procedure
- generic update for linesearch methods is given by
  \[
  x_{k+1} = x_k + \alpha_k p_k
  \]
**Issue 1: steps might be too long**

![Graph](image1)

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!

**What is a practical linesearch?**

- in the “early” days exact linesearches were used, i.e., pick \( \alpha_k \) to minimize \( f(x_k + \alpha_k p_k) \)
  - 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 stepsizes 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

**Definition 2.1 (Steepest descent direction)**

For a differentiable function \( f \), the search direction

\[
p_k \overset{\text{def}}{=} -\nabla f(x_k) \equiv -g_k
\]

is called the steepest-descent direction.

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

\[
\begin{align*}
\text{minimize} & \quad m_k^2(x_k + p) \\
\text{subject to} & \quad \|p\|_2 = \|g_k\|_2
\end{align*}
\]

- \( m_k^2(x_k + p) \overset{\text{def}}{=} f_k + g_k^T p \)
- \( m_k^2(x_k + p) \approx f(x_k + p) \)

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
\[
\min_{p \in \mathbb{R}^n} f_k + g_k^T p + \frac{1}{2} p^T H_k p
\]
- approximates second-derivative Hessian by the identity matrix \( I \)
- how often is this a good idea?
- is it a surprise that convergence is typically very slow?

Idea: choose positive definite \( B \) and compute search direction as the unique minimizer
\[
p_k = \arg\min_{p \in \mathbb{R}^n} m_k^B(p) \triangleq f_k + g_k^T p + \frac{1}{2} p^T B_k p
\]
- \( p_k \) satisfies \( B_k p_k = -g_k \)
- why must \( B_k \) be positive definite?
  - \( B_k \succ 0 \) \( \implies \) \( m_k^B \) is strictly convex \( \implies \) unique solution
  - if \( g_k \neq 0 \), then \( p_k \neq 0 \) and is a descent direction
- \( p_k \) 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 \( s_k \) is the Newton direction. Awesome!

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 \( H = V A V^T \).
4. **If** \( H = 0 \) **then**
5. **Set** \( \varepsilon = 1 \)
6. **Else**
7. **Set** \( \varepsilon = \|H\|_2/\beta > 0 \)
8. **End if**
9. **Compute**
   \[
   \bar{A} = \text{diag}(\bar{\lambda}_1, \bar{\lambda}_2, \ldots, \bar{\lambda}_n) \quad \text{with} \quad \bar{\lambda}_j = \begin{cases} 
\lambda_j & \text{if } \lambda_j \geq \varepsilon \\
\varepsilon & \text{otherwise}
\end{cases}
   \]
10. **Return** \( B = V \bar{A} V^T \succ 0 \), which satisfies \( \text{cond}(B) \leq \beta \)

- replaces the eigenvalues of \( H \) that are “not positive enough” with \( \varepsilon \)
- \( \bar{A} = \Lambda + D \), where
  \[
  D = \text{diag} \left( \max(0, \varepsilon - \lambda_1), \max(0, \varepsilon - \lambda_2), \ldots, \max(0, \varepsilon - \lambda_n) \right) \geq 0
  \]
- \( B = H + E \), where \( E = VDV^T \otimes \varepsilon \)

**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
  \[
  \|A\|_2 = \max_{1 \leq i \leq n} |\lambda_i|
  \]
  with \( \{\lambda_i\} \) the eigenvalues of \( A \).
- the spectral decomposition of \( H \) is given by \( H = VAV^T \), where
  \[
  V = (v_1, v_2, \ldots, v_n) \quad \text{and} \quad A = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)
  \]
  with \( Hv_j = \lambda_j v_j \) and \( \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \).
- \( H \) is positive definite if and only if \( \lambda_j > 0 \) for all \( j \).
- computing the spectral decomposition is, generally, very expensive!

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

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

Taking norms and using the orthogonality of \( V \), gives
\[
\|p\|_2^2 = \|B^{-1}g\|_2^2 = \sum_{j=1}^{n} \left( \frac{v_j^T g}{\bar{\lambda}_j} \right)^2
\]
\[
= \sum_{\lambda_j \geq \varepsilon} \left( \frac{v_j^T g}{\lambda_j} \right)^2 + \sum_{\lambda_j < \varepsilon} \left( \frac{v_j^T g}{\varepsilon} \right)^2
\]

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 \( \varepsilon \to 0 \) 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 ($n < 1000$)

**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** $H = 0$ **then**
   5. Set $\varepsilon = 1$
   6. **else**
      7. Set $\varepsilon = ||H||_2/\beta > 0$
8. **end if**
9. Compute
   
   \[ \bar{A} = \text{diag}(\bar{\lambda}_1, \bar{\lambda}_2, \ldots, \bar{\lambda}_n) \]
   
   with
   \[ \bar{\lambda}_j = \begin{cases} 
   \lambda_j & \text{if } \lambda_j \geq \varepsilon \\
   -\lambda_j & \text{if } \lambda_j \leq -\varepsilon \\
   \varepsilon & \text{otherwise}
   \end{cases} \]

10. **return** $B = V\bar{A}V^T > 0$, which satisfies $\text{cond}(B) \leq \beta$

- replace small eigenvalues $\lambda_i$ of $H$ with $\varepsilon$
- replace “sufficiently negative” eigenvalues $\lambda_i$ of $H$ with $-\lambda_i$
- $\bar{A} = \Lambda + D$, where
  \[ D = \text{diag} \{ \max(0, -2\lambda_1, \varepsilon - \lambda_1), \ldots, \max(0, -2\lambda_n, \varepsilon - \lambda_n) \} \geq 0 \]
- $B = H + E$, where $E = \varepsilon D V^T \succeq 0$

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

\[ E = B - H = V(\bar{A} - A)V^T \]

and since $V$ is orthogonal that

\[ ||E||_2 = ||V(\bar{A} - A)V^T||_2 = ||\bar{A} - A||_2 = \max_j |\bar{\lambda}_j - \lambda_j| \]

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 $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$, 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 $E = 0$ 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

---

**Example 2.2**

Consider

\[ g = \begin{pmatrix} 2 \\ 4 \end{pmatrix} \quad \text{and} \quad H = \begin{pmatrix} 1 & 0 \\ 0 & -2 \end{pmatrix} \]

The Newton direction is

\[ p^N = -H^{-1}g = \begin{pmatrix} -2 \\ 2 \end{pmatrix} \]

so that $g^Tp^N = 4 > 0$ (ascent direction)

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

- Algorithm 1 (Method 1) generates
  \[ B = \begin{pmatrix} 1 & 0 \\ 0 & \varepsilon \end{pmatrix} \]
  
  so that
  \[ p = -B^{-1}g = \begin{pmatrix} -2 \\ -\frac{4}{\varepsilon} \end{pmatrix} \]
  
  and $g^Tp = -4 - \frac{16}{\varepsilon} < 0$ (descent direction)

- Algorithm 2 (Method 2) generates
  \[ B = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} \]

  so that
  \[ p = -B^{-1}g = \begin{pmatrix} -2 \\ -2 \end{pmatrix} \]
  
  and $g^Tp = -12 < 0$ (descent direction)
H = V A V^T

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

\[
A = \begin{pmatrix} A_+ & 0 \\ 0 & A_- \end{pmatrix} \quad \text{and} \quad V = \begin{pmatrix} V_+ \\ V_- \end{pmatrix}
\]

So that \( H = (V_+ \quad V_-) \begin{pmatrix} A_+ & 0 \\ 0 & A_- \end{pmatrix} V_+^T \) and the Newton direction is

\[
p^N = -V A^{-1} V^T g = -(V_+ \quad V_-) \begin{pmatrix} A_+^{-1} & 0 \\ 0 & A_-^{-1} \end{pmatrix} V^T g
\]

\[
= -(V_+ \quad V_-) \begin{pmatrix} A_+^{-1} & 0 \\ 0 & A_-^{-1} \end{pmatrix} \begin{pmatrix} V_+^T g \\ V_-^T g \end{pmatrix}
\]

\[
= -V_+ A_+^{-1} V_+^T g - V_- A_-^{-1} V_-^T g
\]

\[
= -V_+ A_+^{-1} V_+^T g - V_- A_-^{-1} V_-^T g \overset{\text{def}}{=} p_+^N + p_-^N
\]

where

\[
p_+^N = -V_+ A_+^{-1} V_+^T g \quad \text{and} \quad p_-^N = -V_- A_-^{-1} V_-^T g
\]

Proof:

- The function \( \psi(y) \overset{\text{def}}{=} m_0^2(V_+ y) \) may be written as

\[
\psi(y) = g^T V_+ y + \frac{1}{2} y^T V_+^T H_+ y
\]

and has a stationary point \( y^* \) given by

\[
y^* = -(V_+^T H_+)^{-1} V_+^T g
\]

Since

\[
\nabla^2 \psi(y) = V_+^T H_+ V_+ = V_+^T V A^T V_+ = A_+ > 0
\]

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

\[
V_+ y^* = -V_+ A_+^{-1} V_+^T g \equiv p_+^N
\]

- For the second part

\[
y^* = -(V_-^T H_-)^{-1} V_-^T g = -A_-^{-1} V_-^T g
\]

so that

\[
V_- y^* = -V_- A_-^{-1} V_-^T g \equiv p_-^N.
\]

Since \( A_- < 0 \), we know that \( p_-^N \) maximizes \( m_0^2(p) \) on the space spanned by the columns of \( V_- \).
The optimization problem solved in the modified-Newton step satisfies the so-called secant equation

$$B k p = - g k,$$

where

$$A_+ = \{ \lambda_i : \lambda_i \geq \varepsilon \} \quad A_\varepsilon = \{ \lambda_i : |\lambda_i| < \varepsilon \} \quad A_- = \{ \lambda_i : \lambda_i \leq -\varepsilon \}$$

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 $\varepsilon > 0$. 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 $p_k = p_+ + p_\varepsilon + p_-$, where

1. $p_+$ is a direction of positive curvature that minimizes $m_0^k(p)$ in the space spanned by the columns of $V_+$;
2. $-p_-$ is a direction of negative curvature that maximizes $m_0^k(p)$ in the space spanned by the columns of $V_-$;
3. $p_\varepsilon$ is a direction of steepest-descent that maximizes $m_0^k(p)$ in the space spanned by the columns of $V_\varepsilon$.

Comments:
• implies that singular Hessians $H_k$ may easily cause numerical difficulties
• the direction $p_-$ is the negative of $p^\varepsilon$ associated with the Newton step

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

$$\text{minimize}_{p \in \mathbb{R}^n} \quad m_0^{k+1}(p) \overset{\text{def}}{=} f_{x_{k+1}} + g_{x_{k+1}}^T p + \frac{1}{2} p^T B_{k+1} p$$

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

These are equivalent to
• $m_0^{k+1}(0) = f_{x_{k+1}} \quad \checkmark$ already satisfied
• $\nabla m_0^{k+1}(0) = g_{x_{k+1}} \quad \checkmark$ already satisfied
• $\nabla m_0^{k+1}(-\alpha_k p_k) = g_k$

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

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

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

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

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 $B_{k+1} > 0$
• 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 $\alpha_k$ 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.
The problem
Given a symmetric positive-definite matrix $B$, and vectors $s$ and $y$, find a symmetric positive-definite matrix $\tilde{B}$ that satisfies the secant equation

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

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

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

for some vectors $u$ and $v$ that we seek.

**Derivation:**

**Case 1:** $Bs = y$

Choose $u = v = 0$ since

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

**Case 2:** $Bs \neq y$

If $u = 0$ or $v = 0$ then

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

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

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

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

$$\tilde{B} = B + cuu^T$$

**Definition 2.4** (SR1 quasi-Newton formula)

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

$$\tilde{B} = B + \frac{(y - Bs)(y - Bs)^T}{(y - Bs)^T s}$$

and satisfies

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

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

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

to avoid numerical error.

**Question:** Is $\tilde{B}$ positive definite if $B$ is positive definite?

**Answer:** Sometimes...but sometimes not!

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

$$\tilde{B} = B + cuu^T$$

satisfies

$$y \mapsto \tilde{B}s = Bs + cuu^Ts = Bs + \alpha(u^Ts)u$$

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

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

$$= B + \gamma(y - Bs)(y - Bs)^T$$

for some $\gamma \neq 0$ that satisfies the secant equation

$$y \mapsto \tilde{B}s = Bs + \gamma(y - Bs)^T s(y - Bs)$$

which implies

$$y - Bs = \gamma(y - Bs)^T s(y - Bs)$$

which means we can choose

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

**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 $\tilde{B}$ to be positive definite holds. The SR1 update gives

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

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

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

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

Theorem 2.6 (convergence of SR1 update)

Suppose that
- $f$ is $C^2$ on $\mathbb{R}^n$
- $\lim_{k \to \infty} x_k = x^*$ 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
- $|s_k^T(y_k - B_k s_k)| > \kappa \|s_k\| \|y_k - B_k s_k\|_2$ for some $\kappa \in (0, 1)$

Then, the matrices generated by the SR1 formula satisfy

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

Comment: the SR1 updates may converge to an indefinite matrix

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?

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 $\beta$ and $\delta$, 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

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

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 $\beta$ and $\delta$ so that $\bar{B}s = y$ is satisfied. Substituting, we have

$$y \overset{\text{def}}{=} \bar{B}s = (B + \beta Bs(Bs)^T + \delta yy^T)s = Bs + \beta s^T Bs + \delta yy^T s$$

$$= Bs + \beta (s^T Bs)Bs + \delta (y^T s)y$$

Making the “simple” choice of

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

gives

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

$$= Bs - Bs + y = y$$

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

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

(2)

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

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

where

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

Result

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

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

is positive definite and satisfies $B s = y$.

Proof:

Homework assignment. (hint: see the next slide)

Limited-memory BFGS update

Suppose at iteration $k$ we have $m$ previous pairs $(s_i, y_i)$ for $i = k - m, \ldots, k - 1$. 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

$$
B_k = B_k^0 + \sum_{i=k-m}^{k-1} \left[ b, b^T - a, a^T \right]
$$

for some vectors $a$ and $b$ 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

A second derivation of the BFGS formula

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

$$
\text{minimize } ||\tilde{M} - M||_F \quad \text{subject to } \tilde{M} = \tilde{M}^T, \ My = s
$$

for a “closest” symmetric matrix $M$, where

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

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

$$
\tilde{M} = \left( I - \frac{y y^T}{y^T y} \right) M \left( I - \frac{y y^T}{y^T y} \right) + \frac{y y^T}{y^T y}
$$

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

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

(BFGS formula again)

so that $\tilde{B} > 0$ if and only if $M > 0$.

Algorithm 3 Unrolling the L-BFGS formula

1. input $m \geq 1$, $\{ (s_i, y_i) \}$ for $i = k - m, \ldots, k - 1$, and $B_k^0 > 0$
2. for $i = k - m, \ldots, k - 1$ do
3. $b_i \leftarrow y_i / (y_i y_i)^{1/2}$
4. $a_i \leftarrow B_k^0 s_i + \sum_{j=k-m}^{i-1} (B_j^0 y_j b_j - (a_j^0 s_j) b_j)$
5. $a_i \leftarrow a_i / (s_i s_i)^{1/2}$
6. end for

- typically $3 \leq m \leq 30$
- $b_i$ and $b_i^T s_i$ should be saved and reused from previous iterations (exceptions are $b_{k-1}$ and $b_{k-1}^T s_{k-1}$ 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 $\frac{2}{m} n^2$
- cost of a matrix-vector product with $B_k$ requires $4mn$ multiplications.
- other so-called compact representations are slightly more efficient
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

$$\min_{x \in \mathbb{R}^n} q(x) \overset{\text{def}}{=} \frac{1}{2}x^TAx - b^Tx$$

because $\nabla q(x) = Ax - b$ and $\nabla q^T(x) = A > 0$ 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:** $r(x) \overset{\text{def}}{=} Ax - b$ and $r_i \overset{\text{def}}{=} Ax_i - b$.

**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 $A$

$$A = V\Lambda V^T = I\Lambda I^T = \Lambda$$

(a diagonal matrix)

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

$$q(x) = \frac{1}{2}x^TAx - b^Tx$$

**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 $i = 0, 1, \ldots, n - 1$ do
4: Compute $\alpha_i$ as the solution to

$$\min_{\alpha_i \in \mathbb{R}} q(x_i + \alpha_i e_{i+1})$$

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

- $e_i$ represents the $j$th coordinate vector, i.e.,

$$e_j = \begin{pmatrix} 0 & \cdots & 1 & \cdots & 0 \end{pmatrix}^T \in \mathbb{R}^n$$

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

**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 $\tilde{q}$ defined as

$$\tilde{q}(y) \overset{\text{def}}{=} q(Sy) = \frac{1}{2}y^T S^TASy = \frac{1}{2}y^T \hat{A}y$$

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

- If $A = S^TAS$ is diagonal, then applying the coordinate descent Algorithm 4 to $\tilde{q}$ would converge to $y^*$ in $\leq n$ steps.

- $A$ is diagonal if and only if $S^TAs_j = 0$ for all $i \neq j$.

where

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

- 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

$$x_{i+1} = Sy_{i+1} = S(y_i + \alpha_i e_{i+1}) = Sy_i + \alpha_i Se_{i+1} = x_i + \alpha_i s_i.$$
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 $x_k$
4: while $r_k \neq 0$ do
5: Compute $\alpha_k \leftarrow \text{argmin}_{\alpha \in \mathbb{R}} \ q(x_k + \alpha s_k)$
6: Set $x_{k+1} \leftarrow x_k + \alpha s_k$
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

Exercise: show that the $\alpha_k$ 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 $\{s_k\}$, let us look at some properties of them if we assume we already have them.

Theorem 2.7 ($A$-conjugate directions)

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

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

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.

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.

Question: If $\{s_k\}_{k=0}^{n-1}$ 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

$$s_k = -r_k + \beta s_{k-1} \quad \text{for some} \ \beta.$$  \hspace{1cm} (5)

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

$$0 = s_{k-1}^T A s_k$$

$$= s_{k-1}^T A (\beta s_{k-1})$$

$$= s_{k-1}^T A s_{k-1}$$

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

and solving for $\beta$ gives

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

With this choice of $\beta$, 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!
Algorithm 6 Preliminary linear CG algorithm

1: \textbf{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: \hspace{1em} Set $c_k \leftarrow -(s_k^T r_k) / (s_k^T A s_k)$.
5: \hspace{1em} Set $x_{k+1} \leftarrow x_k + c_k s_k$.
6: \hspace{1em} Set $r_{k+1} \leftarrow A x_{k+1} - b$.
7: \hspace{1em} Set $b_{k+1} \leftarrow (s_k^T A r_{k+1}) / (s_k^T A s_k)$.
8: \hspace{1em} Set $s_{k+1} \leftarrow -r_{k+1} + b_{k+1} s_k$.
9: \hspace{1em} Set $k \leftarrow k + 1$.
10: \textbf{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.

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

$$\min_{p \in \mathbb{R}^n} f + g^T p + \frac{1}{2} p^T H p$$

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

Algorithm 8 Newton CG algorithm for computing descent directions

1: \textbf{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: \hspace{1em} if $s_k^T H s_k > 0$ then
5: \hspace{2em} Set $c_k \leftarrow (r_k^T r_k) / (s_k^T H s_k)$.
6: \hspace{2em} else
7: \hspace{3em} if $k = 0$, then return $p_k = -g$ else return $p_k$ end if
8: \hspace{1em} end if
9: \hspace{1em} Set $p_{k+1} \leftarrow p_k + c_k s_k$.
10: \hspace{1em} Set $r_{k+1} \leftarrow r_k + c_k H s_k$.
11: \hspace{1em} Set $b_{k+1} \leftarrow (r_k^T r_{k+1}) / (r_k^T r_k)$.
12: \hspace{1em} Set $s_{k+1} \leftarrow -r_{k+1} + b_{k+1} s_k$.
13: \hspace{1em} Set $k \leftarrow k + 1$.
14: \textbf{end while}
15: \textbf{return} $p_k$

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

Algorithm 7 Linear CG algorithm

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

- Only requires the single matrix-vector multiplication $A s_k$.
- 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

**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 $g^T p_0 = -g^T g = -\|g\|^2 < 0$ 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

$$g^T p_1 = g^T (p_0 + \alpha_0 s_0) = \alpha_0 g^T s_0 = \frac{\alpha_0 = r_0^T g}{s_k^T H s_k} g^T s_0 = \frac{-\|r_0\|^2}{g^T H g} < 0.$$

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 $s_k^T r_k = -\|r_k\|^2$ gives

$$g^T p_{k+1} = g^T p_k + \alpha_k g^T s_k = g^T p_k + \alpha_k s_k^T r_k = g^T p_k + \frac{r_k^T r_k}{s_k^T H s_k} r_k = g^T p_k - \frac{\|r_k\|^2}{s_k^T H s_k} < g^T p_k.$$ 

Combining the previous inequality and (6) results in

$$g^T p_{k+1} < g^T p_k < \cdots < g^T p_1 < -\frac{\|r_1\|^2}{g^T H g} < 0.$$
Algorithm 9 Backtracking

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

- typical choices (not always)
  - $\alpha_{\text{initial}} = 1$
  - $\tau = 1/2$
- 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
  $$f(x_k + \alpha^{(\ell)} p_k) \leq f(x_k)$$

Algorithm 10 A backtracking-Armijo linesearch

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

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

The Armijo condition

Given a point $x_k$ and search direction $p_k$, we say that $\alpha_k$ satisfies the Armijo condition if

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

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.

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 $\alpha \in [0, \alpha_{\text{max}}(x)]$, where

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

Proof:

A Taylor’s approximation and

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

imply that

$$f(x + \alpha p) \leq f(x) + \alpha g(x)^T p + \frac{1}{2} \gamma(x) \alpha^2 \| p \|^2$$

$$\leq f(x) + \alpha g(x)^T p + \alpha (\eta - 1) g(x)^T p$$

$$= f(x) + \alpha \eta g(x)^T p$$
Theorem 3.2

[Bound on $\alpha_k$ when using an Armijo backtracking linesearch] Suppose that
- $f \in C^1$ and $g(x)$ is Lipschitz continuous with Lipschitz constant $\gamma_k$ at $x_k$
- $p_k$ is a descent direction at $x_k$

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

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

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:
- $\alpha_{\text{init}}$ satisfies the Armijo condition: it is then clear that $\alpha_k = \alpha_{\text{init}}$
- $\alpha_{\text{init}}$ 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. \qed

Algorithm 11

```
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 $\alpha_k$ 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 loop
9: return approximate solution $x_k$
```

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 $\gamma$

Then, the iterates generated by the generic backtracking-Armijo linesearch method Algorithm 11 must satisfy one of the following conditions:
- finite termination, i.e., there exists a positive integer $k$ such that $g_k = 0$ for some $k \geq 0$
- objective is unbounded below, i.e.,
  $$\lim_{k \to \infty} f_k = -\infty$$
- an angle condition between $g_k$ and $p_k$ exists, i.e.,
  $$\lim_{k \to \infty} \min \left( \frac{|p_k^T g_k|}{\|p_k\|_2}, \frac{|p_k^T g_k|}{\|p_k\|_2} \right) = 0$$

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

$$\lim_{k \to \infty} f_k > -\infty.$$ \hfill (8)

The Armijo condition implies that

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

for all $k$. Summing over the first $j$ iterations yields

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

Taking limits of the previous inequality as $j \to \infty$ 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

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

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

\[
\sum_{k=0}^{\infty} \alpha_k |p_k^T g_k| \geq \sum_{k=0}^{\infty} \min \left\{ \alpha_{\text{init}}, \frac{2\tau(1-\eta)\|p_k\|_2^2}{\gamma \|p_k\|_2^2} \right\} |p_k^T g_k| \\
= \sum_{k=0}^{\infty} \min \left\{ \alpha_{\text{init}}, \frac{2\tau(1-\eta)|p_k^T g_k|^2}{\gamma \|p_k\|_2^2} \right\} \\
\geq \sum_{k=0}^{\infty} \min \{ \alpha_{\text{init}}, \frac{2\tau(1-\eta)}{\gamma} \min \{ |p_k^T g_k|, \|p_k^T g_k\|_2 \} \}
\]

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

\[
\lim_{k \to \infty} \min \left\{ |p_k^T g_k|, \|p_k^T g_k\|_2 \right\} = 0
\]

\[\phi(\alpha) = f(x_k + \alpha p_k)\]
\[l(\alpha) = f_k + c_1 \alpha g_k^T p_k\]

**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 \( \alpha_k \) satisfies the **Wolfe conditions** if

1. \( f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla(x_k)^T p_k \) (11a)
2. \( \nabla(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla(x_k)^T p_k \) (11b)

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

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 \( \phi(x) := f(x_k + \alpha p_k) \)

**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
  \[ \|\nabla(x_k)\| \leq 10^{-4} \max(1, \|\nabla(x_0)\|) \]
- should also terminate if a maximum number of allowed iterations is reached
- should also terminate if a maximum time limit is reached
Theorem 3.4 (Convergence of generic Wolfe linesearch (Zoutendijk))

Assume that

- \( f : \mathbb{R}^n \to \mathbb{R} \) 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

\[
\sum_{k \geq 0} \cos^2(\theta_k) \|g_k\|^2 < \infty
\]  

(12)

where

\[
\cos(\theta_k) \overset{\text{def}}{=} \frac{-g_k^T p_k}{\|g_k\|_2 \|p_k\|_2}
\]

(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.

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 \( f \)
  - may still allow the acceptance of steps that are far from a univariate minimizer of \( f \)
  - “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)

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 \( \alpha_k \) satisfies the Strong Wolfe conditions if

\[
f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f(x_k)^T p_k
\]

(14a)

\[
|\nabla (x_k + \alpha_k p_k)^T p_k| \leq c_2 |\nabla (x_k)^T p_k|
\]

(14b)

Suppose that \( f \in \mathbb{R}^n \to \mathbb{R} \) 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 \( l(\alpha) := f(x_k) + \alpha_1 \nabla f(x_k)^T p_k \) at least once since \( c_1 \in (0, 1) \); let \( \alpha_m > 0 \) be the smallest such intersecting value so that

\[
f(x_k + \alpha_m p_k) = f(x_k) + \alpha_m \nabla f(x_k)^T p_k.
\]

(15)

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

\[
f(x_k + \alpha_m p_k) - f(x_k) = \alpha_m \nabla f(x_k + \alpha m p_k)^T p_k
\]

for some \( \alpha_m \in (0, \alpha_m) \).

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

\[
\nabla (x_k + \alpha p_k)^T p_k = c_1 \nabla f(x_k)^T p_k > c_2 \nabla f(x_k)^T p_k.
\]

(16)

Thus, \( \alpha_m \) satisfies (11a)/(14a) and (11b) with a strict inequality so we may use the smoothness assumption to deduce that there is an interval around \( \alpha_m \) 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.
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 $(f(x_k + q_k) > f(x_k) + c_1 \cdot t \cdot (g_k^T p_k) \text{ OR } \nabla f(x_k + q_k)^T p_k < c_2 (g_k^T p_k))$ do
4: if $(f(x_k + q_k) > f(x_k) + c_1 \cdot t \cdot (g_k^T p_k) )$ 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: return $t$

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

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

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

$$y_k^T s_k > 0$$

where

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

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 $\alpha_k$, 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 cheaper 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 $\phi$
  - may still allow the acceptance of steps that are far from a univariate minimizer of $\phi$
  - 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 $\phi$
  - only approximate stationary points of the univariate function $\phi$ 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.

Algorithm 14 Steepest descent backtracking-Armijo linesearch method

1: Input: initial guess $x_0$
2: Set $k = 0$
3: until $\|\nabla f(x_k)\| \leq 10^{-6} \max \{1, \|\nabla f(x_0)\|\}$
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
finite termination, i.e., there exists some finite integer \( k \geq 0 \) such that
\[ g_k = 0 \]
objective is unbounded below, i.e.,
\[ \lim_{k \to \infty} f_k = -\infty \]
convergence of gradients, i.e.,
\[ \lim_{k \to \infty} g_k = 0 \]

Proof:
Since \( p_k = -g_k \), if outcome 1 above does not happen, then we may conclude that
\[ \|g_k\| = \|p_k\| = 0 \quad \text{for all} \quad k \geq 0 \]
Additionally, if outcome 2 above does not occur, then Theorem 3.3 ensures that
\[ 0 = \lim_{k \to \infty} \left\{ \min \left( \{ p_k \|p_k\|^2 \|T\| \} / \|p_k\|^2 \right) \right\} = \lim_{k \to \infty} \left\{ \|g_k\| \min (\|g_k\|, 1) \right\} \]
and thus \( \lim_{k \to \infty} g_k = 0. \]

Rate of convergence

Theorem 4.2 (Global convergence rate for steepest-descent)

Suppose \( f \in C^1 \) and \( \nabla f \) 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 \( T \geq 1 \)
\[ \min_{k=0, \ldots, T} \|\nabla f(x_k)\| \leq \frac{M}{\sqrt{T+1}}. \]
Consequently, for any \( \epsilon > 0 \), within \( \left( \frac{2M}{\epsilon^{1/3}} \right) \) steps, we will see an iterate where the gradient has norm at most \( \epsilon \). In other words, we reach an “\( \epsilon \)-stationary” point in \( O\left( \frac{1}{\epsilon^{2/3}} \right) \) 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:
\[ \sum_{k=0}^{r} \alpha_k \|g_k p_k\| \leq \frac{f_0 - f_{k+1}}{\eta} \leq M' \]
since \( f \) is bounded from below.

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 \( O\left( \frac{1}{\epsilon^2} \right) \) to get to a “stationary” point, i.e., \( \|g_k\| \leq \epsilon \)
- 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!
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.

### Algorithm 15 Modified-Newton backtracking-Armijo linesearch method

1. Input: initial guess $x_0$
2. Set $k = 0$
3. until $\|\nabla f(x_k)\| \leq 10^{-8} \max \{1, \|\nabla f(x_k)\|\}$
4. Compute positive-definite matrix $B_k$ from $H_k$ using Algorithm 2.
5. Compute the search direction $p_k$ as the solution to $B_k p_k = -\nabla f_k$.
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

### 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 $\ell$ and upper bounded by $L$.
- The initial starting iterate $x_0$ is close enough to $x^*$; more precisely, $r_0 := \|x_0 - x^*\| < \bar{r} := \frac{\ell}{M}$.
- The steepest-descent with constant step size $\alpha = \frac{1}{2\ell + L}$ converges as follows:
  $$\|x_k - x^*\| \leq \bar{r} r_0 \left(1 - \frac{2\ell}{L + 3\ell}\right)^k.$$  

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

#### 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:

- finite termination, i.e., there exists a finite $k$ such that $g_k = 0$
- objective is unbounded below, i.e., $\lim_{k \to \infty} f_k = -\infty$
- convergence of gradients, i.e., $\lim_{k \to \infty} g_k = 0$

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

$$\lambda_{\min} \leq \lambda_{\min}(B_k) \leq \frac{s^T B_k s}{\|s\|^2} \leq \lambda_{\max}(B_k) \leq \lambda_{\max}$$  

for any nonzero $s$.  

86/106
From the previous slide, it follows that
\[
\lambda_{\max}^{-1} \leq \lambda_{\min}^{-1}(B_k) = \lambda_{\min}(B_k^{-1}) \leq \frac{s^T B_k^{-1} s}{\|s\|^2} \leq \lambda_{\max}(B_k^{-1}) = \lambda_{\min}^{-1}(B_k) \leq \lambda_{\max}^{-1}
\]  
(18)
for any nonzero vector \(s\). Using \(B_k p_k = -g_k\) and (18) we see that
\[
|p_k^T g_k| = ||g_k^T B_k^{-1} g_k| \geq \lambda_{\min}^{-1} \|g_k\|^2
\]  
(19)
In addition, we may use (18) to see that
\[
\|p_k\|^2 = g_k^T B_k^{-2} g_k \leq \lambda_{\max}^{-2} \|g_k\|^2,
\]
which implies that
\[
\|p_k\|^2 \leq \lambda_{\min}^{-1} \|g_k\|^2.
\]
It now follows from the previous inequality and (19) that
\[
\frac{|p_k^T g_k|}{\|p_k\|^2} \geq \frac{\lambda_{\min}^{-1}}{\lambda_{\max}} \|g_k\|^2.
\]
Combining the previous inequality and (19) yields
\[
\min \left( |p_k^T g_k|, \frac{|p_k^T g_k|^2}{\|p_k\|^2} \right) \geq \frac{\|g_k\|^2}{\lambda_{\min}^{-1}} \min \left( 1, \frac{\lambda_{\min}^{-1}}{\lambda_{\max}} \right).
\]
From Theorem 3.3 we know that \(\lim_{k \to \infty} \min \left( |p_k^T g_k|, \frac{|p_k^T g_k|^2}{\|p_k\|^2} \right) = 0\) and combined with (21), we conclude that
\[
\lim_{k \to \infty} g_k = 0.
\]

**Rate of convergence**

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

Suppose \(f \in C^1\) and \(\nabla f\) 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 \(T \geq 1\)
\[
\min_{k=0,\ldots,T} \|\nabla f(x_k)\| \leq \frac{M}{\sqrt{T+1}}.
\]
Consequently, for any \(\epsilon > 0\), within \([\frac{T}{2}, \frac{3T}{2}]\) steps, we will see an iterate where the gradient has norm at most \(\epsilon\). In other words, we reach an "\(\epsilon\)-stationary" point in \(O(\sqrt{T})\) 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.
Mean Value Theorem implies that there exists \( z_k \) between \( x_k \) and \( x_k + p_k \) such that

\[
f(x_k + p_k) = f_k + p_k^Tg_k + \frac{1}{2} p_k^T H(z_k) p_k.
\]

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

\[
f(x_k + p_k) - f_k - \frac{1}{2} p_k^T g_k = \frac{1}{2} p_k^T H(z_k) p_k
= \frac{1}{2} p_k^T (H(z_k) - H_k) p_k
\leq \frac{1}{2} ||H(z_k) - H_k||_2 ||p_k||^2 \leq \frac{1}{2} \gamma ||p_k||^2.
\]

(25)

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

\[
2\gamma ||p_k||^2 \leq \lambda_{\min}(H(x^*))(1 - 2\eta).
\]

(26)

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

\[
f(x_k + p_k) - f_k \leq \frac{1}{2} p_k^T g_k + \frac{1}{2} \gamma ||p_k||^2 \leq \frac{1}{2} p_k^T g_k + \frac{1}{2} \lambda_{\min}(H(x^*))(1 - 2\eta) ||p_k||^2 \leq \frac{1}{2} ||p_k||^2 \leq \eta p_k g_k
\]

so that the unit step \( x_k + p_k \) satisfies the Armijo condition for all \( k \in K \) sufficiently large.

Note that (22) implies that

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

(27)

The iteration gives

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

But a Taylor Approximation provides the estimate

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

which implies

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

(28)

Thus, we know that for \( k \in K \) sufficiently large, the unit step will be accepted and

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

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. \( \blacksquare \)

---

**Algorithm 16** Modified-Newton Wolfe linesearch method

1: Input: initial guess \( x_0 \)
2: Set \( k = 0 \)
3: until \( ||\nabla f_k|| \leq 10^{-\lambda} \max \{ 1, ||\nabla f_k|| \} \) do
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 steplength \( \alpha_k \) 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
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 \( B_0 \).
5: Compute a search direction \( p_k \) as the solution to \( B_k p_k = -g_k \).
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

Theorem 4.7 (Global convergence of modified-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 \( \beta \)

Then, the iterates generated by the modified/quasi-Newton Wolfe Algorithm 17 or 16 satisfy one of the following:
- finite termination, i.e., there exists a finite \( k \) such that \( g_k = 0 \)
- objective is unbounded below, i.e., \( \lim_{k \to \infty} f_k = -\infty \)
- convergence of gradients, i.e., \( \lim_{k \to \infty} g_k = 0 \)

Proof:
Homework assignment.

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

Suppose \( f \in C^1 \) and \( \nabla f \) 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 linesearch Algorithms 16 or 17, such that \( B_k \) has uniformly bounded condition number. Then there exists a constant \( M \) such that for all \( T \geq 1 \)

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

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

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.
The problem

\[ \minimize_{x \in \mathbb{R}^n} \quad f(x) \]

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:

\[ p_k = \arg\min_{p \in \mathbb{R}^n} \quad m_k^2(p) \quad \text{def} = f_k + g_k^T p + \frac{1}{2} p^T B_k p \]

- \( B_k = \) 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.

<table>
<thead>
<tr>
<th>A Summary</th>
<th>A Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Steepest Descent</strong></td>
<td><strong>Modified/Quasi Newton</strong></td>
</tr>
<tr>
<td>Requires only function and gradient evaluations</td>
<td>Could need Hessian evaluations as well (with spectral decompositions)</td>
</tr>
<tr>
<td>No overhead of solving linear system in each iteration</td>
<td>Solves a linear system ( Bp = g ) in each iteration</td>
</tr>
<tr>
<td>Global convergence to ( \epsilon )-stationary point in ( O(\left(\frac{1}{\epsilon}\right)^2) ) steps</td>
<td>Global convergence to ( \epsilon )-stationary point in ( O(\left(\frac{1}{\epsilon}\right)^2) ) steps</td>
</tr>
<tr>
<td>Linear local convergence: converges to ( \epsilon ) distance from local minimum in ( O(\log(\frac{1}{\epsilon})) ) steps</td>
<td>Superlinear (quadratic) local convergence: converges to ( \epsilon ) distance from local minimum in ( O(\log \log(\frac{1}{\epsilon})) ) ( O(\log \log(\frac{1}{\epsilon})) ) steps.</td>
</tr>
</tbody>
</table>