Chapter 3
Gradient-based optimization

Contents (final version)

3.0 Introduction ................................................................. 3.3
3.1 Lipschitz continuity ......................................................... 3.6
3.2 Gradient descent for smooth convex functions ....................... 3.12
3.3 Preconditioned steepest descent ...................................... 3.15
   Preconditioning: overview .............................................. 3.16
   Descent direction ....................................................... 3.17
   Complex case ............................................................ 3.18
3.4 Descent direction for edge-preserving regularizer: complex case .. 3.19
   GD step size with preconditioning .................................... 3.23
   Finite difference implementation ..................................... 3.24
3.5 General inverse problems .............................................. 3.26
3.6 Convergence rates ....................................................... 3.28
   Generalized convergence analysis of PGD ......................... 3.29
   Generalized Nesterov fast gradient method (FGM) ............... 3.31
3.7 First-order methods ...................................................... 3.32

3.1
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>General first-order method classes</td>
<td>3.32</td>
</tr>
<tr>
<td>Optimized gradient method (OGM)</td>
<td>3.36</td>
</tr>
<tr>
<td><strong>3.8 Machine learning via logistic regression for binary classification</strong></td>
<td>3.43</td>
</tr>
<tr>
<td>Adaptive restart of OGM</td>
<td>3.48</td>
</tr>
<tr>
<td><strong>3.9 Summary</strong></td>
<td>3.49</td>
</tr>
</tbody>
</table>
To solve a problem like

\[ \hat{x} = \arg \min_{x \in \mathbb{F}^N} \Psi(x) \]

via an iterative method, we start with some initial guess \( x_0 \), and then the algorithm produces a sequence \( \{x_k\} \) where hopefully the sequence converges to \( \hat{x} \), meaning \( \|x_k - \hat{x}\| \to 0 \) for some norm \( \|\cdot\| \) as \( k \to \infty \).

What algorithm we use depends greatly on the properties of the cost function \( \Psi : \mathbb{F}^N \mapsto \mathbb{R} \).

EECS 551 explored the gradient descent (GD) and preconditioned gradient descent (PGD) algorithms for solving least-squares problems in detail.

Here we review the general form of gradient descent (GD) for convex minimization problems; the LS application is simply a special case.

<table>
<thead>
<tr>
<th>Venn diagram for convex functions:</th>
<th>nonsmooth</th>
<th>composite</th>
<th>differentiable</th>
<th>Lipschitz continuous gradient</th>
<th>twice differentiable with bounded curvature</th>
<th>quadratic (LS)</th>
</tr>
</thead>
</table>
Motivating application(s)

We focus initially on the numerous signal and image processing applications where the cost function is **convex** and **smooth**, meaning it has a **Lipschitz continuous** gradient.

A concrete family of applications is **edge-preserving image recovery** where the measurement model is

\[ y = Ax + \varepsilon \]

for some matrix \( A \) and we estimate \( x \) using

\[
\hat{x} = \arg \min_{x \in \mathbb{F}^N} \frac{1}{2} \| Ax - y \|_2^2 + \beta R(x)
\]

where the regularizer is convex and smooth, such as

\[
R(x) = \sum_{k} \psi([Cx]_k)
\]

for a potential function \( \psi \) that has a Lipschitz continuous derivative, such as the **Fair potential**.
Example. Here is an example of **image deblurring** or **image restoration** that was performed using such a method. The left image is the blurry noisy image $y$, and the right image is the restored image $\hat{x}$. 
3.1 Lipschitz continuity

The concept of **Lipschitz continuity** is defined for general metric spaces, but we focus on vector spaces.

Define. A function $g : \mathbb{F}^N \mapsto \mathbb{F}^M$ is **Lipschitz continuous** if there exists $L < \infty$, called a **Lipschitz constant**, such that

$$\|g(x) - g(z)\| \leq L \|x - z\|, \quad \forall x, z \in \mathbb{R}^N.$$ 

In general the norms on $\mathbb{F}^N$ and $\mathbb{F}^M$ can differ and $L$ will depend on the choice of the norms. We will focus on the Euclidean norms unless otherwise specified.

Define. The smallest such $L$ is called the **best Lipschitz constant**. (Often just “the” LC.)

**Algebraic properties**

Let $f$ and $g$ be Lipschitz continuous functions with (best) Lipschitz constants $L_f$ and $L_g$ respectively.

<table>
<thead>
<tr>
<th>$h(x)$</th>
<th>$L_h$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha f(x) + \beta$</td>
<td>$</td>
</tr>
<tr>
<td>$f(x) - x_0$</td>
<td>$L_f$</td>
</tr>
<tr>
<td>$f(x) + g(x)$</td>
<td>$\leq L_f + L_g$</td>
</tr>
<tr>
<td>$f(g(x))$</td>
<td>$\leq L_f L_g$</td>
</tr>
<tr>
<td>$Ax + b$</td>
<td>$|A|_2$</td>
</tr>
<tr>
<td>$f(x)g(x)$</td>
<td>?</td>
</tr>
</tbody>
</table>
If \( f \) and \( g \) are Lipschitz continuous functions on \( \mathbb{R} \), then \( h(x) = f(x)g(x) \) is a Lipschitz continuous function on \( \mathbb{R} \). (?)

A: True  B: False

If \( f : \mathbb{F}^N \mapsto \mathbb{F} \) and \( g : \mathbb{F}^N \mapsto \mathbb{F} \) are Lipschitz continuous functions and \( h(x) \triangleq f(x)g(x) \), and \( |f(x)| \leq f_{\text{max}} < \infty \) and \( |g(x)| \leq g_{\text{max}} < \infty \), then \( h(\cdot) \) is Lipschitz continuous on \( \mathbb{F}^N \) and \( L_h \leq f_{\text{max}}L_g + g_{\text{max}}L_f \).

Proof

\[
|h(x) - h(z)| = |f(x)g(x) - f(z)g(z)| \\
= |f(x)(g(x) - g(z)) - (f(z) - f(x))g(z)| \\
\leq |f(x)| |g(x) - g(z)| + |g(z)| |f(x) - f(z)| \text{ by triangle ineq.} \\
\leq (f_{\text{max}}L_g + g_{\text{max}}L_f) \|x - z\|_2
\]

Is boundedness of both \( f \) and \( g \) a necessary condition?

No. Think \( f(x) = \alpha \) and \( g(x) = x \). Then \( L_{fg} = |\alpha| = f_{\text{max}} \) but \( g(\cdot) \) is unbounded yet Lipschitz.
For our purposes, we especially care about cost functions whose gradients are Lipschitz continuous. We call these smooth functions. The definition of gradient is subtle for functions on $\mathbb{C}^N$ so here we focus on $\mathbb{R}^N$.

Define. A differentiable function $f(x)$ has a **Lipschitz continuous gradient** if $\exists L < \infty$ such that

$$
\|\nabla f(x) - \nabla f(z)\|_2 \leq L \|x - z\|_2, \quad \forall x, z \in \mathbb{R}^N.
$$

Example. For $f(x) = \frac{1}{2} \|Ax - y\|_2^2$ we have

$$
\|\nabla f(x) - \nabla f(z)\|_2 = \|A'(Ax - y) - A'(Az - y)\|_2
$$

$$
= \|A'Ax - z\|_2 \leq \|A'A\|_2 \|x - z\|_2.
$$

So the Lipschitz constant of $\nabla f$ is $L = \|A'A\|_2 = \|A\|_2^2 = \sigma^2(A) = \rho(A'A)$.

See [8, p. 56] for many equivalent conditions for a differentiable function $f$ to have a Lipschitz continuous gradient, such as

$$
f(x) \geq f(z) + \langle \nabla f(z), x - z \rangle, \quad \forall x, z \in \mathbb{R}^N \quad \text{tangent plane property}
$$

$$
f(x) \leq f(z) + \langle \nabla f(z), x - z \rangle + \frac{L}{2} \|x - z\|_2^2, \quad \forall x, z \in \mathbb{R}^N \quad \text{quadratic majorization property}.
$$

The first property holds for all differentiable convex functions.
Fact. If \( f(x) \) is twice differentiable and if there exists \( L < \infty \) such that its Hessian matrix has a bounded spectral norm:
\[
\|\nabla^2 f(x)\|_2 \leq L, \quad \forall x \in \mathbb{R}^N,
\]
then \( f(x) \) has a Lipschitz continuous gradient with Lipschitz constant \( L \).

So twice differentiability with bounded curvature is sufficient, but not necessary, for a function to have Lipschitz continuous gradient.

Proof. Using Taylor's theorem and the triangle inequality and the definition of spectral norm:
\[
\|\nabla f(x) - \nabla f(z)\|_2 = \left\| \int_0^1 \nabla^2 f(x + \tau(z - x)) \, d\tau \, (x - z) \right\|_2 \\
\leq \left( \int_0^1 \|\nabla^2 f(x + \tau(z - x))\|_2 \, d\tau \right) \|z - z\|_2 \\
\leq \left( \int_0^1 L \, d\tau \right) \|x - z\|_2 = L \|x - z\|_2.
\]

Example. \( f(x) = \frac{1}{2} \|Ax - y\|_2^2 \implies \nabla^2 f = A' A \) so \( \|\nabla^2 f\|_2 = \|A' A\|_2 = \|A\|_F^2. \)

Example. The Lipschitz constant for the gradient of \( f(x) \triangleq x' \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} x \) is:
\( \nabla^2 f = 2zz' \) where \( z' = [1 \ 2] \) so \( \|\nabla^2 f\|_2 = 2 \|z\|^2_2 = 10. \)
Example. The **Fair potential** used in many imaging applications [1] [2] is

\[ \psi(z) = \delta^2 \left( |z/\delta| - \log(1 + |z/\delta|) \right) , \]

(3.1)

for some \( \delta > 0 \) and has the property of being roughly quadratic for \( z \approx 0 \) and roughly like \( |z| \) for \( \delta |z| \gg 0 \). When the domain of \( \psi \) is \( \mathbb{R} \), we can differentiate (carefully treat \( z > 0 \) and \( z < 0 \) separately):

\[ \dot{\psi}(z) = \frac{z}{1 + |z/\delta|} \quad \text{and} \quad \ddot{\psi}(z) = \frac{1}{(1 + |z/\delta|)^2} \leq 1 , \]

so the Lipschitz constant of the derivative of \( \psi(\cdot) \) is 1. Furthermore, its second derivative is nonnegative so it is a convex function.

In the figure, \( \delta = 1 \).

Example. Is the Fair potential \( \psi \) itself **Lipschitz continuous**? Yes: \( |\dot{\psi}(z)| \leq 1 \).
Edge-preserving regularizer and Lipschitz continuity

Example. Determine “the” Lipschitz constant for the gradient of the edge-preserving regularizer in $\mathbb{R}^N$, when the derivative $\dot{\psi}$ of potential function $\psi$ has Lipschitz constant $L_{\dot{\psi}}$:

$$R(x) = \sum_{k=1}^K \psi([Cx]_k) \implies \nabla R(x) = \mathbf{C}' \dot{\psi}.(\mathbf{C}x) = h(g(f(x))),$$

(3.2)

where $f(x) = \mathbf{C}x$, $g(u) = \dot{\psi}.(u)$, $h(v) = \mathbf{C}'v$, $L_f = \|\mathbf{C}\|_2$, $L_h = \|\mathbf{C}'\|_2$. (students finish it)

$$\|g(u) - g(v)\|_2^2 = \|\dot{\psi}.(u) - \dot{\psi}.(v)\|_2^2 = \sum_k \|\dot{\psi}(u_k) - \dot{\psi}(v_k)\|_2^2 \leq \sum_k L_{\dot{\psi}}^2 |u_k - v_k|^2 = L_{\dot{\psi}}^2 \|u - v\|_2^2$$

$$\implies L_g = L_{\dot{\psi}},$$

$$\implies L_{\nabla R} = L_{\dot{\psi}} \|\mathbf{C}\|_2^2 = L_{\dot{\psi}} \|\mathbf{C}'\|_2^2.$$

(3.3)

Thus when $\ddot{\psi} \leq 1$, the Lipschitz constant of the gradient of the above $R(x)$ is:

A: 1       B: $\|\mathbf{C}\|_2$       C: $\|\mathbf{C}'\|_2$       D: $\|\mathbf{C}\|_2^4$       E: None of these    ??
3.2 Gradient descent for smooth convex functions

- If convex function $\Psi(x)$ has a (not necessarily unique) minimizer $\hat{x}$ for which
  \[ -\infty < \Psi(\hat{x}) \leq \Psi(x), \quad \forall x \in \mathbb{R}^N, \]

- the gradient of $\Psi(x)$ is **Lipschitz continuous**, i.e.,
  \[ \|\nabla \Psi(x) - \nabla \Psi(z)\|_2 \leq L \|x - z\|_2, \quad \forall x, z \in \mathbb{R}^N, \]

- the step size $\alpha$ is chosen such that
  \[ 0 < \alpha < 2/L, \]

then the GD iteration
\[ x_{k+1} = x_k - \alpha \nabla \Psi(x_k) \]

has the following convergence properties [3, p. 207].

- The cost function is non-increasing (monotone): $\Psi(x_{k+1}) \leq \Psi(x_k), \quad \forall k \geq 0.$
- The distance to any minimizer $\hat{x}$ is non-increasing (monotone): $\|x_{k+1} - \hat{x}\|_2 \leq \|x_k - \hat{x}\|_2, \quad \forall k \geq 0.$
- The sequence $\{x_k\}$ converges to a minimizer of $\Psi(\cdot)$.
- The gradient norm tends to zero [3, p. 22] $\|\nabla \Psi(x_k)\|_2 \to 0$.
- For $0 < \alpha \leq 1/L$, the cost function decrease is bounded by [4]:
  \[ \Psi(x_k) - \Psi(\hat{x}) \leq \frac{L\|x_0 - \hat{x}\|_2^2}{2} \max \left( \frac{1}{2k\alpha + 1}, (1 - \alpha)^2k \right). \]

This upper bound is conjectured to also hold for $1/L < \alpha < 2/L$ [5].
Convergence rates

There are many ways to assess the convergence rate of an iterative algorithm like GD. Researchers study:

- \( \Psi(x_k) \rightarrow \Psi(\hat{x}) \)
- \( \|\nabla \Psi(x_k)\| \rightarrow 0 \)
- \( \|x_k - \hat{x}\| \rightarrow 0 \)
  both globally and locally...

Quantifying bounds on the rates of decrease of these quantities is an active research area. Even classical GD has relatively recent results \[4\] that tighten up the traditional bounds. The tightest possible worst-case bound for GD for the decrease of the cost function (with a fixed step size \( \alpha = 1/L \)) is \( O(1/k) \):

\[
\Psi(x_k) - \Psi(\hat{x}) \leq \frac{\|x_0 - \hat{x}\|^2}{L(4k + 2)},
\]

where \( L \) is the Lipschitz constant of the gradient \( \nabla f(x) \).

In contrast, Nesterov’s fast gradient method (p. 3.31) has a worst-case cost function decrease at rate at least \( O(1/k^2) \), which can be improved (and has) by only a constant factor \[6\].
Example. The following figure illustrates how slow GD can converge for a simple LS problem with \( A = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \) and \( y = 0 \). This case used the optimal step size \( \alpha_* \) for illustration. This slow convergence has been the impetus of thousands of papers on faster algorithms!

The ellipses show the contours of the LS cost function \( \| Ax - y \| \).

Two ways to try to accelerate convergence are to use a **preconditioner** and/or a **line search**.
3.3 Preconditioned steepest descent

Instead of using GD with a fixed step size $\alpha$, an alternative is to do a **line search** to find the best step size at each iteration. This variation is called **steepest descent** (or GD with a line search). Here is how **preconditioned steepest descent** for a linear LS problem works:

- **Search direction** (negative preconditioned gradient):
  \[ d_k = -P \nabla \Psi(x_k) \]

- **Step size**:
  \[ \alpha_k = \arg \min_{\alpha} \Psi(x_k + \alpha d_k) \]

- **Update**:
  \[ x_{k+1} = x_k + \alpha_k d_k \]

- Finding $\alpha_k$ analytically quadratic cases is a HW problem.
- By construction, this iteration is guaranteed to decrease the cost function monotonically: $\Psi(x_{k+1}) \leq \Psi(x_k)$ with strict decrease unless $x_k$ is already a minimizer, provided the preconditioner $P$ is positive definite.
- Computing $\alpha_k$ takes some extra work, especially for non-quadratic problems. Often Nesterov’s fast gradient method or the optimized gradient method (OGM) [6] are preferable because they do not require a line search (if the Lipschitz constant is available).
3.16

Preconditioning: overview

Why use $-P \nabla \Psi(x)$ as a search direction?

Consider $\Psi(x) = \frac{1}{2} \|Ax - y\|^2_2$ and define $f(z) \triangleq \Psi(Tz) = \frac{1}{2} \|ATz - y\|^2_2$. The Gram matrix of $f(\cdot)$ is $\nabla^2 f(z) = T'A'AT$.

Applying GD to $f$ yields

$$z_{k+1} = z_k - \alpha T'A'(A'Tz_k - y)$$
$$\Rightarrow Tz_{k+1} = Tz_k - \alpha TT'A'(A'Tz_k - y)$$
$$\Rightarrow x_{k+1} = x_k - \alpha PA'(A'x_k - y) = x_k - \alpha P \nabla \Psi(x_k),$$

where $x_k \triangleq Tz_k$ and $P = TT'$. So ordinary GD on $f$ is the same as preconditioned GD on $\Psi$.

If $\alpha = 1$ and $T = (A'A)^{-1/2}$, then $P = (A'A)^{-1}$ and $x_1 = (A'A)^{-1}A'y$.

In this sense $P = (A'A)^{-1} = [\nabla^2 \Psi(x)]^{-1}$ is the ideal preconditioner.
Descent direction

Define. A vector \( d \in \mathbb{F}^N \) descent direction for a cost function \( \Psi : \mathbb{F}^N \mapsto \mathbb{R} \) at a point \( x \) iff moving locally from \( x \) along the direction \( d \) decreases the cost, i.e.,

\[
\exists \ c = c(x, d, \Psi) > 0 \ \text{s.t.} \ \forall \epsilon \in [0, c) \ \Psi(x + \epsilon d) \leq \Psi(x).
\] (3.4)

With this definition \( d = 0 \) is always a (degenerate) descent direction.

Fact. For \( \mathbb{R}^N \), if \( \Psi(x) \) is differentiable at \( x \) and \( P \) is positive definite, then the following vector, if nonzero, is a descent direction for \( \Psi \) at \( x \):

\[
d = -P \nabla \Psi(x).
\] (3.5)

Proof sketch. Taylor’s theorem yields

\[
\Psi(x) - \Psi(x + \alpha d) = -\alpha \langle \nabla \Psi(x), d \rangle + o(\alpha) = \alpha \left[ d' P^{-1} d + \frac{o(\alpha)}{\alpha} \right],
\]

which will be positive for sufficiently small \( \alpha \), because \( d' P^{-1} d > 0 \) for \( P > 0 \), and \( o(\alpha)/\alpha \to 0 \) as \( \alpha \to 0 \).

From this analysis we can see that designing/selecting a preconditioner that is positive definite is crucial. The two most common choices are:

- \( P \) is diagonal with positive diagonal elements,
- \( P = QDQ' \) where \( D \) is diagonal with positive diagonal elements and \( Q \) is unitary.

In this case \( Q \) is often circulant so we can use FFT operations to perform \( Pg \) efficiently.
Complex case

The definition of **descent direction** in (3.4) is perfectly appropriate for both $\mathbb{R}^N$ and $\mathbb{F}^N$. However, the direction $d$ specified in (3.5) is problematic in general on $\mathbb{F}^N$ because many cost functions of interest are not **holomorphic** so are not differentiable on $\mathbb{F}^N$.

However, despite not being differentiable, we can still find a **descent direction** for most cases of interest.

**Example.** The most important case of interest here is $\Psi : \mathbb{C}^N \mapsto \mathbb{R}$ defined by $\Psi(x) = \frac{1}{2} \| Ax - y \|^2_2$ where $A \in \mathbb{C}^{M \times N}$ and $y \in \mathbb{C}^M$. This function is not **holomorphic**. However, one can show that

$$d = -PA'(Ax - y)$$

is a **descent direction** for $\Psi$ at $x$ when $P$ is a positive definite matrix. (HW)

In the context of optimization problems, when we write $g = \nabla \Psi(x) = A'(Ax - y)$ for the complex case, we mean that $-g$ is a descent direction for $\Psi$ at $x$, not a derivative.

Furthermore, one can show (HW) that the set of minimizers of $\Psi(x)$ is the same as the set of points that satisfy $\nabla \Psi(x) = A'(Ax - y) = 0$. So again even though a derivative is not defined here, the descent direction sure walks like a duck and talks like a **duck**, I mean like a (negated) derivative.
3.4 Descent direction for edge-preserving regularizer: complex case

Now consider an edge-preserving regularizer defined on $\mathbb{C}^N$:

$$R(x) = \sum_{k=1}^{K} \psi([Cx]_k), \text{ where } \psi(z) = f(|z|)$$

(3.6)

for some some potential function $\psi : \mathbb{C} \mapsto \mathbb{R}$ defined in terms of some function $f : \mathbb{R} \mapsto \mathbb{R}$.

If $f(r) = r^2$ then it follows from p. 3.18 that $C'Cx$ is a descent direction for $R(x)$ on $\mathbb{C}^N$.

But it seems unclear in general how to define a descent direction, due to the $|\cdot|$ above.

To proceed, make the following assumptions about the function $f$:

• $f : \mathbb{R} \mapsto \mathbb{R}$.
• $0 \leq s \leq t \Rightarrow f(s) \leq f(t)$ (monotone).
• $f$ is differentiable on $\mathbb{R}$.
• $\omega_f(t) \triangleq \frac{\dot{f}(t)}{t}$ is well-defined for all $t \in \mathbb{R}$, including $t = 0$.
• $0 \leq \omega_f(t) \leq \omega_{\text{max}} < \infty$.

Example. For the Fair potential $\omega_f(t) = 1/ (1 + |t/\delta|) \in (0, 1]$. 
Claim. For these assumptions, a descent direction for the edge-preserving regularizer (3.6) for $x \in \mathbb{F}^N$ is

$$-\nabla R(x) = -C' \text{diag}\{\omega_f.(|Cx|)\} Cx,$$

where the $|\cdot|$ is evaluated element-wise, like $\text{abs}().$ in JULIA. Intuition: $\dot{f}(t) = \omega_f(t) \cdot t.$ To prove this claim, we focus on just one term in the sum:

$$r(x) \triangleq \psi(v'x) = f(|v'x|), \quad d(x) = -v \omega_f(|v'x|) v'.$$

for some nonzero vector $v \in \mathbb{C}^N.$ (A row of $C'$ in (3.6) corresponds to $v'$.)

Letting $\omega = \omega_f(|v'x|) \leq \omega_{\text{max}}$ we have

$$r(x + \epsilon d) = f(|v'(x + \epsilon d)|) = f(|v'(x - \epsilon \omega vv'x)|) = f(|v'x(1 - \epsilon \omega v'v)|) = f(|v'x| 1 - \epsilon \omega ||v||^2).$$

Now when $0 \leq \epsilon \leq 1/(\omega ||v||^2)$ then

$$1 - \epsilon \omega ||v||^2 \in [0, 1],$$

$$\Rightarrow r(x + \epsilon d) = f(|v'x| 1 - \epsilon \omega ||v||^2) \leq f(|v'x|) = r(x),$$

using the monotone property of $f(\cdot).$ So $d(x)$ is a descent direction for $r(x).$

The proof of (3.7) involves the sum of $K$ such terms.
So with our usual reuse of $\nabla$ to denote a (negated) descent direction, we will not write (3.2) for $\mathbb{C}^N$ but rather we will define $\omega_{\psi} = \omega_f$ and write:

$$\nabla R(x) = C' \text{diag}\{\omega_{\psi}.(|Cx|)\} Cx.$$  

If $\psi(z) = \psi(|z|), \forall z \in \mathbb{C}$, then we can define $\omega_{\psi}$ in terms of $\dot{\psi}$ for nonnegative real arguments.

**Lipschitz constant for descent direction**

Having established the descent direction (3.7) for edge-preserving regularization on $\mathbb{C}^N$, the next step is to determine a Lipschitz constant for that function.

Again we can write it as a composition of three functions:

$$C' \text{diag}\{\omega_{\psi}.(|Cx|)\} Cx = h(g(f(x))), \quad f(x) = Cx, \quad g(u) = d(u), \quad h(v) = C'v, \quad d(z) \triangleq \omega_{\psi}(|z|) z.$$  

For real $z$ arguments and when $\dot{\psi}$ and hence $\omega_{\psi}$ are symmetric, $d(z) = \dot{\psi}(z)$ so the Lipschitz constant is easy. When $z \in \mathbb{C}$, I have not yet been able to prove Lipschitz continuity of $d(z)$. My conjecture, supported by numerical experiments, is that if $\omega_{\psi}(t)$ is a non-increasing function of $t$ on $[0, \infty)$, in addition to the other assumptions on p. 3.19, then

$$L_d = \omega_{\psi}(0)$$

and, akin to (3.3), the Lipschitz constant for the descent direction on $\mathbb{C}^N$ is:

$$L_{\nabla R} = \|C\|_2^2 L_d.$$  

(3.8)
**Practical Lipschitz constant**

In general, computing $\|C\|_2^2$ in (3.3) exactly would require a SVD or the power iteration, both of which are impractical for large-scale problems.

If we use finite differences with periodic boundary conditions, then $C$ is circulant and hence is a **normal matrix** so $\|C\|_2 = \rho(C)$, where $\rho(\cdot)$ denotes the **spectra radius**. For 1D finite differences with $N$ even the spectral radius is 2 and for $N$ odd is $1 + \cos(\pi(N - 1)/N) \approx 2$. (HW)

But for nonperiodic boundary conditions we need a different approach.

Recall that because $C'C$ is symmetric:

$$\|C\|_2^2 = \|C'C\|_2 = \sigma_1(C'C) = \rho(C'C) \leq \|C'C\|,$$

for any matrix norm $\|\cdot\|$. In particular, the matrix 1-norm is convenient:

$$\|C'C\|_1 \leq \|C'\|_1 \|C\|_1 = \|C\|_\infty \|C\|_1 \Rightarrow \|C\|_2^2 \leq \|C\|_\infty \|C\|_1 = 2 \cdot 2 = 4$$

for 1D finite differences, because there as most a single $+1$ and $-1$ in each row or column of $C$.

Interestingly, this 1-norm approach gives us an upper bound on $\|C\|_2^2$ for any boundary conditions that matches the exact value when using periodic boundary conditions.

So the practical choice for 1D first-order finite-differences is to use $L_{\nabla R} = 4L_{\psi}$, where often we scale the potential functions so that $L_{\psi} = 1$. Bottom line: never use `opnorm()` when working with finite differences!
GD step size with preconditioning

We earlier argued that the preconditioned gradient $-P \nabla \Psi(x)$ can be preferable to $-\nabla \Psi(x)$ as a descent direction. But the GD convergence theorem on p. 3.12 had no $P$ in it. So must we resort to PSD on p. 3.15 that requires a line search? No!

Suppose $\Psi$ is convex has a Lipschitz continuous gradient with Lipschitz constant $L_{\nabla \Psi}$. Define a new function in transformed coordinate system: $f(z) \triangleq \Psi(Tz)$. Using the properties on p. 3.6, this function also has a Lipschitz continuous gradient and

$$\nabla f(z) = T' \nabla \Psi(Tz) \implies L_{\nabla f} \leq \|T'T\|_2 L_{\nabla \Psi}.$$  

Choose a step size $0 < \alpha_f < 2/L_{\nabla f}$ for applying GD to $f$, yielding

$$z_{k+1} = z_k - \alpha_f \nabla f(z_k) = z_k - \alpha_f T' \nabla \Psi(Tz_k) \implies Tz_{k+1} = Tz_k - \alpha_f TT' \nabla \Psi(Tz_k) \implies x_{k+1} = x_k - \alpha_f P \nabla \Psi(x_k),$$

where $x_k \triangleq Tz_k$ and $P = TT'$. So ordinary GD on $f$ is the same as preconditioned GD on $\Psi$. The step size should satisfy $0 < \alpha_f < 2/L_{\nabla f}$ so it suffices (but can be suboptimal; see HW) to choose

$$0 < \alpha_f < \frac{2}{\|T'T\|_2 L_{\nabla \Psi}} = \frac{1}{\|P\|_2 L_{\nabla \Psi}} 2.$$
Finite difference implementation

For \( x \in \mathbb{F}^N \) we need to compute first-order finite differences \( d_n = x_{n+1} - x_n, \ n = 1, \ldots, N - 1, \) which in matrix notation is \( d = Cx \in \mathbb{F}^{N-1}. \) Here are eight (!) different implementations in JULIA:

```julia
function loopdiff(x::AbstractVector)
    N = length(x)
    y = similar(x, N-1)
    for n=1:(N-1)
        @inbounds y[n] = x[n+1] - x[n]
    end
    return y
end

d = diff(x) # built-in

d = loopdiff(x)

d = (circshift(x, -1) - x)[1:(N-1)]

d = [x[n+1]-x[n] for n=1:(length(x)-1)] # comprehension

d = @views x[2:end] - x[1:(end-1)] # indexing

d = conv(x, eltype(x).([1, -1]))[2:end-1] # using DSP

d = diagm(0 => -ones(N-1), 1 => ones(N-1))[1:(N-1),:] * x # big/slow

d = spdiagm(0 => -ones(N-1), 1 => ones(N-1))[1:(N-1),:] * x # SparseArrays
```
Which is fastest? https://web.eecs.umich.edu/~fessler/course/598/demo/diff1.html

Use notebook to discuss
@inbounds
@views
spdiags (column-wise storage...)
LinearMaps (cf fmatrix in MIRT)

**Adjoint tests**

\[ y'(Ax) = (A'y)'x \] or equivalently \[ \langle Ax, y \rangle = \langle x, A'y \rangle \]

```julia
x = randn(N); y = randn(M);
@assert isapprox(y'*(A*x), (A'*y)'*x)
```

A generalization of transpose for linear maps is called the **adjoint**.
3.5 General inverse problems

As mentioned previously, a typical cost function for solving inverse problems has the form

$$\hat{x} = \arg \min_{x \in \mathbb{F}^N} \frac{1}{2} \|Ax - y\|_2^2 + \beta R(x), \quad R(x) = \sum_{k=1}^K \psi([Cx]_k) = r(Cx), \quad r(v) = \sum_{k=1}^K \psi(v_k).$$  \hspace{1cm} (3.9)

This is just one of many possible special cases of the following fairly general form

$$\Psi(x) = \sum_{j=1}^J f_j(B_j x) \implies \nabla \Psi(x) = \sum_{j=1}^J B_j' \nabla f_j(B_j x),$$  \hspace{1cm} (3.10)

where $B_j$ is a $M_j \times N$ matrix and each $f_j : \mathbb{R}^{M_j} \mapsto \mathbb{R}$ is a (typically convex) function.

**Example.** For the special case (3.9), we use (3.10) with

$$J = 2, \quad B_1 = A, \quad B_2 = C, \quad f_1(u) = \frac{1}{2} \|u - y\|_2^2, \quad f_2(v) = \beta r(v).$$

We will implement several algorithms for minimize cost functions of this general form.

When $\psi$ is the Fair potential, the (best) Lipschitz constant of $\nabla f_2(v)$ is:

A: 1  B: $\beta$  C: $\beta \sqrt{K}$  D: $\beta K$  E: None  ??
Efficient line search

A naive implementation of the line search step in the PSD algorithm on p. 3.15 minimizes

$$h_k(\alpha) \triangleq \Psi(x_k + \alpha d_k).$$

When applied to a cost function of the general form (3.10) this would involve repeated matrix-vector multiplications of the form $B_j(x_k + \alpha d_k)$, which is expensive. A more efficient approach is to precompute the matrix-vector products prior to performing the line search, noting that for the general (3.10) form:

$$h_k(\alpha) = \Psi(x_k + \alpha d_k) = \sum_{j=1}^{J} f_j(B_j(x_k + \alpha d_k)) = \sum_{j=1}^{J} f_j(u_j^{(k)} + \alpha v_j^{(k)}), \quad u_j^{(k)} \triangleq B_j x_k, \quad v_j^{(k)} \triangleq B_j d_k.$$

Precomputing $u_j^{(k)}$ and $v_j^{(k)}$ prior to performing the line search avoids redundant matrix-vector products. Furthermore, algorithms with line searches like PSD and PCG have an recursive update of the form:

$$x_{k+1} = x_k + \alpha_k d_k.$$

Multiplying both sides by $B_j$ yields an efficient recursive update for the $u_j$ vector (used in HW problems):

$$B_j x_{k+1} = B_j x_k + \alpha_k B_j d_k \implies u_j^{(k+1)} = u_j^{(k)} + \alpha v_j^{(k)}.$$

A key simplification here is that the Lipschitz constant of $h_k$ does not use operator norms of any $B_j$. (HW)
3.6 Convergence rates

Asymptotic convergence rates

When the cost function $\Psi$ is locally strictly convex and twice differentiable near minimizer $\hat{x}$, one can analyze the asymptotic convergence rates of PGD, PSD, and PCG. (See Fessler book Ch. 11.)

All three algorithms satisfy inequalities of the following form for different values of $c, \rho$:

$$
\left\| P^{-1/2}(x_{k+1} - \hat{x}) \right\|_2 \leq c \rho^k \left\| P^{-1/2}(x_0 - \hat{x}) \right\|_2 \implies \lim_{k \to \infty} \left\| P^{-1/2}(x_k - \hat{x}) \right\|_2^{1/k} \leq \rho.
$$

PGD and PSD produce sequences $\{x_k\}$ that converge linearly [8, p. 32] to $\hat{x}$ and

$$
\sup_{x_0} \lim_{k \to \infty} \left\| P^{-1/2}(x_k - \hat{x}) \right\|_2^{1/k} = \rho,
$$

where $\rho$ is called the root convergence factor. Define $\hat{H} = P^{1/2} \nabla^2 \Psi(\hat{x}) P^{1/2}$ and condition number $\kappa = \sigma_1(\hat{H}) / \sigma_N(\hat{H})$.

This table shows the values of $\rho$.

<table>
<thead>
<tr>
<th>Method</th>
<th>$\rho$</th>
<th>$\kappa = 10^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGD standard step $\alpha = \frac{1}{\sigma_1(\hat{H})}$</td>
<td>$\frac{\kappa - 1}{\kappa}$</td>
<td>0.99</td>
</tr>
<tr>
<td>PGD $\alpha_* = \frac{2}{\sigma_1(\hat{H}) + \sigma_N(\hat{H})}$</td>
<td>$\frac{\kappa - 1}{\kappa + 1}$</td>
<td>0.98</td>
</tr>
<tr>
<td>PSD with perfect line search</td>
<td>$\frac{\kappa - 1}{\kappa + 1}$</td>
<td>0.98</td>
</tr>
<tr>
<td>PCG with perfect line search</td>
<td>$\frac{\sqrt{\kappa - 1}}{\sqrt{\kappa} + 1}$</td>
<td>0.82</td>
</tr>
</tbody>
</table>

PCG converges quadratically [8, p. 45] and its $\rho$ above matches a lower bound [8, p. 68].
Generalized convergence analysis of PGD

Define. The gradient of $\Psi$ is **$S$-Lipschitz continuous** on $\mathbb{R}^N$ for an invertible matrix $S$ iff

$$\| S^{-1} (\nabla \Psi(x) - \nabla \Psi(z)) \|_2 \leq \| S' (x - z) \|_2 , \quad \forall x, z \in \mathbb{R}^N. \quad (3.11)$$

If $S = \sqrt{L} I$, then the $S$-Lipschitz condition (3.11) simplifies to the classic Lipschitz continuity condition:

$$\| \nabla \Psi(x) - \nabla \Psi(z) \|_2 \leq L \| x - z \|_2 , \quad \forall x, z \in \mathbb{R}^N. \quad (3.12)$$

**Theorem** (PGD convergence). If $\nabla \Psi$ satisfies (3.11) and for some $0 < \alpha$

$$\alpha P' SS' P \prec P + P', \quad (3.13)$$

then (i) the PGD algorithm

$$x_{k+1} = x_k - \alpha P \nabla \Psi(x_k) \quad (3.14)$$

monotonically decreases $\Psi$ [7] because for $g = \nabla \Psi(x_k)$:

$$\Psi(x_k) - \Psi(x_{k+1}) \geq \frac{\alpha}{2} g' (P + P' - \alpha P' SS' P) g,$$

and (ii) $\| \nabla \Psi(x_k) \| \to 0$. 
In the usual case where $P$ is symmetric positive definite, (3.13) simplifies to $\alpha S S' \prec 2P^{-1}$ and when that condition holds then $\|P^{-1/2}(x_k - \hat{x})\|_2$ converges monotonically to zero [7].

If we choose $\alpha P = (S S')^{-1}$, then PGD is equivalent to a majorize-minimize (MM) method (discussed later) and the cost function decrease has the following bound:

$$\Psi(x_k) - \Psi(\hat{x}) \leq \frac{\|S'(x_0 - \hat{x})\|^2}{2k}, \quad k \geq 1.$$  \hfill (3.15)

The bound above is the “classical” textbook formula. In 2014 the following tight bound was found: [4]:

$$\Psi(x_k) - \Psi(\hat{x}) \leq \frac{\|S'(x_0 - \hat{x})\|^2}{4k + 2}, \quad k \geq 1.$$  \hfill (3.16)

It is tight because there is a Huber-like function $\Psi$ for which GD meets that rate.

**Why generalize? Units of Lipschitz constant**

Consider $\Psi(x) = \frac{1}{2} \|Ax - y\|^2$ where $A$ is $2 \times 2$ diagonal with units:

<table>
<thead>
<tr>
<th>$a_{11}$</th>
<th>ampere</th>
<th>$x_1$</th>
<th>ohm</th>
<th>$y_1$</th>
<th>volt</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_{22}$</td>
<td>volt/m</td>
<td>$x_2$</td>
<td>m</td>
<td>$y_2$</td>
<td>volt</td>
</tr>
</tbody>
</table>

What are the units of the Lipschitz constant of $\nabla \Psi$?

A: ampere volt / m   B: ampere$^2$   C: volt$^2$ / m$^2$   D: ohm m   E: none of these

??
Generalized Nesterov fast gradient method (FGM)

The following is a slight generalization of the fast gradient method (FGM) of Nesterov, also known as accelerated gradient descent and Nesterov accelerated gradient (NAG):

Initialize $t_0 = 1$ and $z_0 = x_0$ then for $k = 0, 1, \ldots$:

$$t_{k+1} = \left(1 + \sqrt{1 + 4t_k^2}\right) / 2$$

$$x_{k+1} = z_k - [SS']^{-1} \nabla \Psi(z_k) \quad \text{gradient step}$$

$$z_{k+1} = x_{k+1} + \frac{t_k - 1}{t_{k+1}} (x_{k+1} - x_k) \quad \text{momentum}$$

If $t_k = 1$ for all $k$ then FGM reverts to ordinary GD.

**Theorem** If $\Psi$ is convex and has an $S$-Lipschitz gradient, then this generalized FGM satisfies:

$$\Psi(x_k) - \Psi(\hat{x}) \leq 2 \frac{\|SS' (x_0 - \hat{x})\|^2}{k^2}.$$  \hspace{1cm} (3.17)

In words, the worst-case rate of decrease of the cost function $\Psi$ is $O(1/k^2)$.

This is a huge improvement over the $O(1/k)$ rate of PGD in (3.15).

However, worst-case analysis may be pessimistic for your favorite application. For example, if $\Psi$ is quadratic on $\mathbb{R}^N$, then CG converges in $N$ iterations.
### 3.7 First-order methods

The most famous second-order method is Newton’s method:
\[
x_{k+1} = x_k - (\nabla^2 \Psi(x_k))^{-1} \nabla \Psi(x_k).
\]

This method is impractical for large-scale problems so we focus on first-order methods [8, p. 7].

**General first-order method classes**
- General first-order (GFO) method:
  \[
x_{k+1} = \text{function}(x_0, \Psi(x_0), \nabla \Psi(x_0), \ldots, \Psi(x_k), \nabla \Psi(x_k)).
  \]
  (3.18)
- First-order (FO) methods with fixed step-size coefficients:
  \[
x_{n+1} = x_n - \frac{1}{L} \sum_{k=0}^{n} h_{n+1,k} \nabla \Psi(x_k).
  \]
  (3.19)

Which of the algorithms discussed so far are FO (fixed-step) methods?
A: PGD, FGM, PSD, PCG  B: PGD, FGM, PSD  C: PGD, PSD  D: PGD, FGM  E: PGD  ??
Example: Barzilai-Borwein gradient method

Barzilai & Borwein, 1988: [9]

\[
\begin{align*}
g_k & \triangleq \nabla \Psi(x_k) \\
\alpha_k &= \frac{\|x_k - x_{k-1}\|^2}{\langle x_k - x_{k-1}, g_k - g_{k-1} \rangle} \\
x_{k+1} &= x_k - \alpha_k g_k.
\end{align*}
\]

- In “general” first-order (GFO) class, but
- not in class FO with fixed step-size coefficients.

Recent research questions

- Analyze convergence rate of FO for any given step-size coefficients \(\{h_{n,k}\}\)
- Optimize step-size coefficients \(\{h_{n,k}\}\)
  - fast convergence
  - efficient recursive implementation
  - universal (design prior to iterating, independent of \(L\))
- How much better could one do with GFO?
Nesterov’s fast gradient method is FO

Nesterov (1983) iteration \([10, 11]\) expressed in efficient recursive form: Initialize: \( t_0 = 1, \ z_0 = x_0 \)

\[
\begin{align*}
  z_{n+1} &= x_n - \frac{1}{L} \nabla \Psi(x_n) \quad \text{(usual GD update)} \\
  t_{n+1} &= \frac{1}{2} \left( 1 + \sqrt{1 + 4t_n^2} \right) \quad \text{(magic momentum factors)} \\
  x_{n+1} &= z_{n+1} + \frac{t_n - 1}{t_{n+1}} (z_{n+1} - z_n) \quad \text{(update with momentum)}.
\end{align*}
\]

FGM1 is in class FO \([6]\) (for analysis, not implementation!):

\[
\begin{align*}
  x_{n+1} &= x_n - \frac{1}{L} \sum_{k=0}^{n} h_{n+1,k} \nabla \Psi(x_k) \\
  h_{n+1,k} &= \begin{cases} \\
    \frac{t_n - 1}{t_{n+1}} h_{n,k}, & k = 0, \ldots, n - 2 \\
    \frac{t_n - 1}{t_{n+1}} (h_{n,n-1} - 1), & k = n - 1 \\
    1 + \frac{t_n - 1}{t_{n+1}}, & k = n.
  \end{cases}
\end{align*}
\]

\[
\begin{bmatrix}
  1 & 0 & 0 & 0 & 0 & 0 \\
  0 & 1.25 & 0 & 0 & 0 & 0 \\
  0 & 0.10 & 1.40 & 0 & 0 & 0 \\
  0 & 0.05 & 0.20 & 1.50 & 0 & 0 \\
  0 & 0.03 & 0.11 & 0.29 & 1.57 & 0 \\
  0 & 0.02 & 0.07 & 0.18 & 0.36 & 1.62
\end{bmatrix}
\]
Nesterov’s FGM1 optimal convergence rate

Shown by Nesterov to be $O(1/n^2)$ for “primary” sequence $\{z_n\}$:

\[
\Psi(z_n) - \Psi(x_\star) \leq \frac{2L \|x_0 - x_\star\|^2}{(n+1)^2}.
\]  

(3.20)

Nesterov [12, p. 59-61] constructed a simple quadratic function $\Psi$, with a tridiagonal Hessian matrix similar to $C'C$ for first-order differences, such that, for any general FO method:

\[
\frac{3}{32} L \|x_0 - x_\star\|^2 \leq \Psi(x_n) - \Psi(x_\star).
\]

Thus the $O(1/n^2)$ rate of FGM1 is “optimal” in a big-O sense.

Bound on convergence rate of “secondary” sequence $\{x_n\}$ [6]:

\[
\Psi(x_n) - \Psi(x_\star) \leq \frac{2L \|x_0 - x_\star\|^2}{(n+2)^2}.
\]

To reach a cost within $\varepsilon$ of the minimum $\Psi(x_\star)$, how many iterations are needed?

A: $O(1)$  B: $O(1/\sqrt{\varepsilon})$  C: $O(1/\varepsilon)$  D: $O(1/\varepsilon^2)$  E: $O(1/\varepsilon^4)$

The gap between 2 and 3/32 suggests we can do better, and we can, thanks to recent work from UM.
Optimized gradient method (OGM)

Recall general family of first-order (FO) methods (3.19) with fixed step-size coefficients:

\[ x_{n+1} = x_n - \frac{1}{L} \sum_{k=0}^{n} h_{n+1,k} \nabla \Psi(x_k). \]

Inspired by [4], recent work by former UM ECE PhD student Donghwan Kim [6]:

- Analyze (i.e., bound) convergence rate as a function of
  - number of iterations \( N \)
  - Lipschitz constant \( L \)
  - step-size coefficients \( H = \{h_{n+1,k}\} \)
  - initial distance to a solution: \( R = \|x_0 - x_*\| \).

- Optimize \( H \) by minimizing the bound. “Optimizing the optimizer” (meta-optimization?)

- Seek an equivalent recursive form for efficient implementation.
(... many pages of derivations ...)

- Optimized step-size coefficients [6]:

\[
H^* : \ h_{n+1,k} = \begin{cases} 
\frac{\theta_n - 1}{\theta_{n+1}} h_{n,k} , & k = 0, \ldots, n - 2 \\
\frac{\theta_n - 1}{\theta_{n+1}} (h_{n,n-1} - 1) , & k = n - 1 \\
1 + \frac{2\theta_n - 1}{\theta_{n+1}} , & k = n.
\end{cases}
\]

\[
\theta_n = \begin{cases} 
1 , & n = 0 \\
\frac{1}{2} \left( 1 + \sqrt{1 + 4\theta_n^2} \right) , & n = 1, \ldots, N - 1 \\
\frac{1}{2} \left( 1 + \sqrt{1 + 8\theta_n^2} \right) , & n = N.
\end{cases}
\]

- Analytical convergence bound for FO method with these optimized step-size coefficients:

\[
\Psi(x_N) - \Psi(x_*) \leq \frac{1L \|x_0 - x_*\|^2_2}{(N + 1)(N + 1 + \sqrt{2})} \leq \frac{1L \|x_0 - x_*\|^2_2}{(N + 1)^2}.
\]

- Of course bound is \(O(1/N^2)\), but constant is twice better than that of Nesterov’s FGM in (3.20).
Optimized gradient method (OGM) recursion

Donghwan Kim also found an efficient recursive algorithm [6]:
Initialize: $\theta_0 = 1$, $z_0 = x_0$

\[
\begin{align*}
    z_{n+1} &= x_n - \frac{1}{L} \nabla \Psi(x_n) \\
    \theta_n &= \begin{cases} 
        \frac{1}{2} \left( 1 + \sqrt{1 + 4\theta_{n-1}^2} \right), & n = 1, \ldots, N - 1 \\
        \frac{1}{2} \left( 1 + \sqrt{1 + 8\theta_{n-1}^2} \right), & n = N 
    \end{cases} \\
    x_{n+1} &= z_{n+1} + \frac{\theta_n - 1}{\theta_{n+1}} (z_{n+1} - z_n) + \frac{\theta_n}{\theta_{n+1}} (z_{n+1} - x_n).
\end{align*}
\]

Reverts to Nesterov’s FGM if one removes the new term.
- Very simple modification of existing Nesterov code.
- Factor of 2 better bound than Nesterov’s “optimal” FGM.
- Similar momentum to Güler’s 1992 proximal point algorithm [13].
- Inconvenience: must pick $N$ in advance.
Recent refinement of OGM

New version OGM’ [14]:

\[ z_{n+1} = x_n - \frac{1}{L} \nabla \Psi(x_n) \]

\[ t_{n+1} = \frac{1}{2} \left( 1 + \sqrt{1 + 4t_n^2} \right) \quad \text{(momentum factors)} \]

\[ x_{n+1} = x_n - \frac{1 + t_n/t_{n+1}}{L} \nabla \Psi(x_n) + \frac{t_n - 1}{t_{n+1}} (z_{n+1} - z_n). \]

- New convergence bound for every iteration:

\[ \Psi(z_n) - \Psi(x^*) \leq \frac{1L \|x_0 - x^*\|_2^2}{(n + 1)^2}. \]

- Simpler and more practical implementation.
- Need not pick \( N \) in advance.
OGM’ momentum factors illustrated

\[ x_{n+1} = x_n - \frac{1 + t_n/t_{n+1}}{L} \nabla \Psi(x_n) + \frac{t_n - 1}{t_{n+1}} (z_{n+1} - z_n) \]

Intuition:
\[ 1 + t_n/t_{n+1} \to 2 \text{ as } n \to \infty \]
Optimized gradient method (OGM) is an optimal GFO method (!) 

Recall that within the class of first-order (FO) methods with fixed step sizes:

$$x_{n+1} = x_n - \frac{1}{L} \sum_{k=0}^{n} h_{n+1,k} \nabla \Psi(x_k),$$

OGM is based on optimized \( \{h_{n,k}\} \) step sizes and provides the convergence rate upper bound:

$$\Psi(x_N) - \Psi(x_*) \leq \frac{L \|x_0 - x_*\|^2}{N^2}.$$ 

Recently Y. Drori [15] considered the class of general FO (GFO) methods:

$$x_{n+1} = F(x_0, \Psi(x_0), \nabla \Psi(x_0), \ldots, \Psi(x_n), \nabla \Psi(x_n)),$$

and showed for \( d > N \) (large-scale problems), any algorithm in this GFO class has a function \( \Psi \) such that

$$\frac{L \|x_0 - x_*\|^2}{N^2} \leq \Psi(x_N) - \Psi(x_*),$$

Thus OGM has optimal (worst-case) complexity among all GFO methods, not just fixed-step FO methods.
Worst-case functions for OGM

From [14], worst-case behavior of OGM is for a **Huber** function and a **quadratic** function.

For $R \triangleq \|x_0 - x^*_\|$, the worst-case behavior is:

$$\Psi(x_N) - \Psi(x^*) = \frac{LR^2}{\theta^2_N} \leq \frac{LR^2}{(N + 1)(N + 1 + \sqrt{2})} \leq \frac{LR^2}{(N + 1)^2}.$$

**Monotonicity**

In these examples, the cost function $\Psi(x_n)$ happens to decrease monotonically. In general, neither FGM nor OGM guarantee non-increasing cost functions, despite the bound $1/N^2$ being strictly decreasing. Nesterov [8, p. 71] states that optimal methods in general do not ensure that $\Psi(x_{k+1}) \leq \Psi(x_k)$. 
3.8 Machine learning via logistic regression for binary classification

To learn weights $\mathbf{x} \in \mathbb{R}^N$ of a binary classifier given feature vectors $\{\mathbf{v}_i\} \in \mathbb{R}^N$ (training data) and labels $\{y_i = \pm 1 : i = 1, \ldots, M\}$, we can minimize a cost function with a regularization parameter $\beta > 0$:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \Psi(\mathbf{x}), \quad \Psi(\mathbf{x}) = \sum_{i=1}^{M} \psi(y_i \langle \mathbf{x}, \mathbf{v}_i \rangle) + \beta \frac{1}{2} \|\mathbf{x}\|_2^2.$$  \hfill (3.21)

Want:
- $\langle \mathbf{x}, \mathbf{v}_i \rangle > 0$ if $y_i = +1$
- $\langle \mathbf{x}, \mathbf{v}_i \rangle < 0$ if $y_i = -1$,
- i.e., $\langle \mathbf{x}, y_i \mathbf{v}_i \rangle > 0$,

so that $\text{sign}(\langle \mathbf{x}, \mathbf{v}_i \rangle)$ is a reasonable classifier.

**Logistic** loss function has a Lipschitz derivative:

$$\psi(z) = \log(1 + e^{-z})$$
$$\dot{\psi}(z) = \frac{-1}{e^z + 1}$$
$$\ddot{\psi}(z) = \frac{e^z}{(e^z + 1)^2} \in \left(0, \frac{1}{4}\right].$$
The logistic regression cost function (3.21) is a special case of the “general inverse problem” (3.10). (?)

A: True  B: False

\[ J = 2, \quad B_1 = V', \quad B_2 = I, \quad f_1(u) = 1' \psi(u), \quad f_2(v) = \beta \frac{1}{2} ||v||^2_2 \]

\[ V = [y_1v_1 \ldots y_Mv_M] \in \mathbb{R}^{M \times N}. \]

A regularization term like \( \frac{\beta}{2} ||x||^2_2 \) is especially important in the typical case where the feature vector dimension \( N \) is large relative to the sample size \( M \).

For gradient-based optimization, we need the cost function gradient and a Lipschitz constant:

\[ \nabla \Psi(x) = V \dot{\psi}(V'x) + \beta x \implies L \nabla \Psi \leq ||V||^2_2 L \dot{\psi} + \beta = \frac{1}{4} ||V||^2_2 + \beta. \]

Practical implementation:
- Normalizing each column of \( V \) to unit norm can help keep \( e^x \) from overflowing.
- Tuning \( \beta \) should use cross validation or other such tools from machine learning.
- The cost function is convex with Lipschitz gradient, so it is well-suited for FGM and OGM.
- When feature dimension \( N \) is very large, seeking a sparse weight vector \( x \) may be preferable. For that, replace the Tikhonov regularizer \( ||x||^2_2 \) with \( ||x||_1 \) and then use FISTA (or POGM [16]) for optimization.
Numerical Results: logistic regression

Labeled training data (green and blue points);
initial decision boundary (red);
final decision boundary (magenta);
ideal boundary (yellow).
$M = 100$, $N = 7$ (cf “large scale” ?)
Numerical Results: convergence rates

OGM faster than FGM in early iterations...
Adaptive restart of accelerated GD

FGM restart, O’Donoghue & Candès, 2015 [18]
OGM restart [19]
Adaptive restart of OGM

Recall:

\[ x_{n+1} = x_n - \frac{1 + t_n/t_{n+1}}{L} \nabla \Psi(x_n) + \frac{t_n - 1}{t_{n+1}} (z_{n+1} - z_n) \]

Heuristic: restart momentum (set \( t_n = 1 \)) if

\[ \langle -\nabla \Psi(x_n), z_{n+1} - z_n \rangle < 0. \]

This modified method, OGM-restart, has a better worst-case convergence bound than OGM for the class of convex cost functions with \( L \)-Lipschitz smooth gradients. (??)

A: True  
B: False

Define. A function \( f : \mathbb{R}^N \mapsto \mathbb{R} \) is strongly convex with parameter \( \mu > 0 \) iff it is convex and

\[ f(x) \geq f(z) + \langle \nabla f(z), x - z \rangle + \frac{1}{2\mu} \|x - z\|^2, \quad \forall x, z \in \mathbb{R}^N. \]

Smooth cost functions are often locally strongly convex, but rarely are the cost functions of interest in modern signal processing (globally) strongly convex.

Formal analysis of OGM for strongly convex quadratic functions is in [19].

Code: https://gitlab.eecs.umich.edu/michigan-fast-optimization/ogm-adaptive-restart
3.9 Summary

This chapter summarizes some of the most important gradient-based algorithms for solving unconstrained optimization problems with differentiable cost functions.

All of the methods discussed here require computing the gradient $\nabla \Psi(x_k)$ each iteration, and often that is the most expensive operation.

Some of the algorithms (PSD, PCG) also require a line search step. A line search is itself a 1D optimization problem that requires evaluating the cost function or its gradient multiple times, and those evaluations can add considerable expense for general cost functions.

For cost functions of the form (3.10), where each component function $f_j$ and its gradient are easy to evaluate, one can perform a line search quite efficiently, as described on p. 3.27.

The set of cost functions of the form (3.10), where each $f_j$ has a Lipschitz continuous gradient, is a strict subset of the set of cost functions $\Psi$ having Lipschitz continuous gradients. (?)

A: True  B: False

Bibliography