Chapter 11
Optimization by General-Purpose Methods

Contents

11.1 Introduction (s.opt,intro) ............................................. 11.3
11.1.1 Iterative optimization methods .................................. 11.3
11.1.2 The Hessian .................................................. 11.3
11.1.3 Why so many algorithms? ...................................... 11.4
11.1.4 Monotonicity (s,opt,mono) .................................... 11.4
* General-purpose unconstrained optimization methods ............... 11.5
11.2 Fixed-point iterations (s.opt,fixed) ................................ 11.5
11.3 Preconditioned gradient descent (PGD) algorithms (s,opt,pgd) .... 11.5
11.3.1 Monotonicity conditions for PGD ................................ 11.5
11.3.2 Convergence in norm ........................................... 11.7
11.3.3 Local convergence rate (s,opt,pgd,rate) ........................ 11.7
11.3.4 Convergence rate of cost function decrease: $O(1/n)$ (s,opt,pgd,1n) 11.8
11.3.5 Relationship with optimization transfer (s.opt,pgd,ox) .......... 11.9
11.3.6 Step-halving or backtracking (s.opt,pgd,half) .................. 11.9
11.3.7 Ideal preconditioner (s.opt,precon) ............................ 11.9
11.3.8 Preconditioning as a coordinate transformation .................. 11.10
11.4 Newton-Raphson algorithm (s,opt,nr) ............................... 11.10
11.5 Preconditioned steepest descent (PSD) algorithms (s,opt,psd) .... 11.10
11.5.1 Orthogonality and search directions ............................ 11.11
11.5.2 Complex case .............................................. 11.11
11.5.3 Asymptotic convergence rate ................................... 11.11
11.6 Line search methods (s,opt,line) .................................... 11.12
11.6.1 Line search using Lipschitz conditions (s,opt,line,lips) ......... 11.12
11.6.2 Step size using Newton’s method (s.opt,line,newt) ............. 11.12
11.6.3 Line search for PWLS cost functions (s.opt,line,pwls) ......... 11.13
11.7 Accelerated first-order methods (s,opt,a1) .......................... 11.13
11.7.1 Barzilai and Borwein gradient method (s.opt,bb) ............... 11.13
11.7.2 Nesterov’s “optimal” first-order methods (s.opt,nesterov) .... 11.14
11.8 Preconditioned conjugate gradients (PCG) (s,opt,pcg) ............... 11.15
11.8.1 Asymptotic convergence rate ................................... 11.17
11.9 Quasi-Newton methods (s,opt,qn) ................................. 11.17
11.9.1 Implementation ............................................. 11.18
* General-purpose constrained optimization methods ................. 11.19
11.10 Gradient projection method (s,opt,gp) ............................. 11.19
11.10.1 Example (relation to SQS) (s,opt,gp,sqs) ..................... 11.20
11.10.2 Limitations to preconditioning (s.opt,gp,pre) .................. 11.21
11.11 Newton’s method (s,opt,con,nr) .................................. 11.21
11.12 Conjugate gradient method (s,opt,con,cg) ......................... 11.22
11.13 Coordinate descent methods (s,opt,cd) .............................. 11.22
11.13.1 Coordinate-descent Newton (CD-NR)-Raphson ................. 11.23
11.13.2 Asymptotic convergence rate of CD ........................... 11.23
11.13.3 Block coordinate descent .................................... 11.24
11.14 Other constrained optimization methods (s,opt,con,other) ........................................ 11.25
  11.14.1 Variable transformation methods ................................................................. 11.25
  11.14.2 Penalty methods ......................................................................................... 11.25
* Special-purpose optimization methods ............................................................... 11.26
11.15 Projection onto convex sets (POCS) (s,opt,pocs) ................................................ 11.26
  11.15.1 POCS algorithm (sequential) ................................................................. 11.26
  11.15.2 POCS algorithm (parallel projections) ...................................................... 11.26
  11.15.3 Discussion ......................................................................................... 11.27
11.16 ART and its relatives (s,opt,art1) .................................................................... 11.27
  11.16.1 ART ................................................................................................. 11.27
  11.16.2 MART ............................................................................................. 11.27
  11.16.3 SART .............................................................................................. 11.28
  11.16.4 SIRT ................................................................................................. 11.28
  11.16.5 Parallel projection version of SIRT .............................................................. 11.29
  11.16.6 SIRT for quadratically regularized WLS ...................................................... 11.29
11.17 ART2 (s,opt,art2) ......................................................................................... 11.30
11.18 Annealing methods (s,opt,anneal) ..................................................................... 11.31
  11.18.1 Simulated annealing ............................................................................. 11.31
  11.18.2 Deterministic annealing and graduated nonconvexity ............................. 11.31
  11.18.3 Other global optimization methods .......................................................... 11.31
11.19 Problems (s,opt,prob) .................................................................................... 11.31
11.20 Bibliography .................................................................................................. 11.33
11.1 Introduction (s.opt.intro)

Most of the image reconstruction algorithms described in subsequent chapters of this book are variations of a modest set of basic iterative methods. This chapter describes some of those basic methods in general terms. It should be helpful to simply browse this chapter as preparation for subsequent chapters, and then return here to the specific descriptions for details as needed later. Entire books have been devoted to optimization e.g., [1–5], and survey papers [6–8]. We focus here on the families of algorithms that have been applied to image reconstruction problems, or that lend insight into such algorithms.

Throughout this chapter, our goal is to find a minimizer \( \hat{x} \) of a cost function \( \Psi(x) \), where \( \Psi : \mathbb{R}^{n_p} \to \mathbb{R} \). We make two general assumptions about \( \Psi \) throughout this chapter:

1. \( \Psi \) is differentiable
   \[ \text{Assumption 1: } \Psi \text{ is differentiable} \]  

2. \( \Psi \) has a finite global minimizer, i.e.,
   \[ \exists \hat{x} \in \mathbb{R}^{n_p} : -\infty < \Psi(\hat{x}) \leq \Psi(x), \quad \forall x \in \mathbb{R}^{n_p}. \]  

Optimization of non-smooth cost functions is more complicated. Assuming that \( \Psi \) has a finite minimizer excludes functions like \( e^x \). Even though this particular example is a strictly convex function (see § 27.9.3), it is an undesirable cost function. Fortunately, the cost functions of interest in imaging usually satisfy the above assumptions.

11.1.1 Iterative optimization methods

Mathematically, we are interested primarily in one or both of the following two problems:

\[ \hat{x} = \arg \min_x \Psi(x) \]  

or the nonnegativity-constrained version

\[ \hat{x} = \arg \min_{x \succeq 0} \Psi(x). \]

In other words, our goal is to find \( \hat{x} \) such that \( \Psi(\hat{x}) \leq \Psi(x), \forall x \succeq 0. \)

In the absence of any constraints such as nonnegativity, a minimizer \( \hat{x} \) of a differentiable cost function \( \Psi(x) \) is necessarily [9, p. 178] a solution of the following system of \( n_p \) equations in \( n_p \) unknowns:

\[ \nabla \Psi(x) \bigg|_{x=\hat{x}} = 0, \]

where \( \mathbf{0} \) denotes the \( n_p \times 1 \) vector of zeros, and \( \nabla \) denotes the column gradient vector:

\[ \nabla \Psi(x) \triangleq \begin{bmatrix} \frac{\partial}{\partial x_1} \Psi(x) \\ \vdots \\ \frac{\partial}{\partial x_{n_p}} \Psi(x) \end{bmatrix}. \]

For most of the problems of interest in image reconstruction, there are no closed-form solutions to the system of equations (11.1.5), even if we disregard the nonnegativity constraint. And in the few cases (such as weighted least squares problems, see Chapter 14) where closed-form solutions exist, those direct solutions are generally computationally intractable (usually involving the inverse of large matrices). Thus an iterative algorithm is required for optimizing \( \Psi \) to find \( \hat{x} \).

11.1.2 The Hessian

Some of the algorithms described below use the Hessian of the cost function \( \Psi \), either in implementation or in analysis. For a twice-differential cost function \( \Psi \), the Hessian matrix

\[ H(x) \triangleq \nabla^2 \Psi(x) \]

has \( j, k \)th element given by

\[ h_{jk}(x) = \frac{\partial^2}{\partial x_j \partial x_k} \Psi(x), \quad k, j = 1, \ldots, n_p. \]

For later convenience, we denote the Hessian at the \( n \)th iteration as follows:

\[ H_n(x^{(n)}) \]  

---

1 One context in which non-smooth optimization arises in imaging problems is when the cost function involves the absolute value function, such as in total variation methods discussed in § 24. One often approximates \( |x| \) with \( \sqrt{x^2 + \epsilon} \) for some small positive \( \epsilon \), thereby ensuring differentiability. There are also methods for non-smooth optimization problems, such as those involving the \( 1 \)-norm, as discussed in § 1.12.3.

2 Actually, it suffices for \( \Psi \) to have a finite minimizer over the set of feasible parameters when constraints such as nonnegativity are involved.

3 The notation \( \arg \min_x \) indicates the (or an) argument \( x \) that minimizes the subsequent expression.
11.1.3 Why so many algorithms?

When developing algorithms for image reconstruction, there are many design considerations, most of which are common to any problem involving iterative methods. In particular, an algorithm designer should consider the impact of design choices on the following characteristics.

- Convergence rate (as few iterations as possible)
- Computation time per iteration (as few floating point operations as possible)
- Nonnegativity constraint ($x \geq 0$)
- Parallelization
- Sensitivity to numerical errors
- Storage requirements (as little memory as possible)
- Memory bandwidth (data access)
- Ease of implementation and code maintenance

These are often conflicting requirements, and one must make compromises appropriate for a given application.

11.1.4 Monotonicity

Along with the properties listed above, there is an additional important property that often plays a significant role in proving algorithm convergence: monotonicity. We say an algorithm is monotone if it generates a sequence $\{x^{(n)}\}$ that decreases $\Psi$ each iteration, i.e., if

$$
\Psi(x^{(n+1)}) \leq \Psi(x^{(n)}), \quad n = 0, 1, 2, \ldots
$$

Such algorithms generally have desirable convergence properties, and are analyzed in detail in Chapter 13.

**Definition 11.1.1** An algorithm is strictly monotone if and only if $\Psi(x^{(n+1)}) < \Psi(x^{(n)})$ for all $x^{(n)}$ that are not minimizers, i.e., such that $\min_x \Psi(x) < \Psi(x^{(n)})$.

Monotonicity sometimes comes at the price of conservative convergence rates compared to some more aggressive (but occasionally nonmonotone) method. A useful hybrid approach can be to use a monotone method as a “fallback” update when a more aggressive optimization method fails to decrease the cost function on a given iteration.

Strict monotonicity alone does not ensure that a sequence converges, even if one also shows that the gradient vanishes in the limit, i.e., $\nabla \Psi(x^{(n)}) \to 0$.

**Example 11.1.2** Consider the convex and differentiable cost function $\Psi(x) = (|x| - 1)^2 \mathbb{1}_{\{|x| \geq 1\}}$. The sequence $x^{(n)} = (-1)^n (1 + 1/n)$ has the strictly monotone property $\Psi(x^{(n+1)}) < \Psi(x^{(n)})$ and derivative $\Psi(x^{(n)}) \to 0$, but $\{x^{(n)}\}$ does not converge. For more subtle examples of such pathologies see [10, 11].
General-purpose unconstrained optimization methods

Unconstrained minimization is the simplest case, and many general-purpose unconstrained optimization methods have been applied to image reconstruction problems.

11.2 Fixed-point iterations (s,opt,fixed)

Multiplying both sides of the necessary condition (11.1.5) by some \( n_p \times n_p \) matrix \( M(x) \) of the algorithm designer’s choice (that in fact may or may not depend on \( x \)), and then subtracting both sides from \( \hat{x} \) yields the equality

\[
\hat{x} = \hat{x} - M(\hat{x}) \nabla \Psi(\hat{x}).
\]

(11.2.1)

The evocative form of this equality suggests the following family of iterative algorithms:

\[
x^{(n+1)} = x^{(n)} - M(x^{(n)}) \nabla \Psi(x^{(n)}). \tag{11.2.2}
\]

Algorithms that are “derived” by replacing an equality like (11.2.1) with a recursion like (11.2.2) are called fixed-point iterations or successive substitution methods. Only in special circumstances will such algorithms be globally convergent\(^4\) [9, p. 272], although they sometimes are locally convergent\(^5\). Despite this significant disadvantage, several fixed-point iterations appear in the literature.

For specific problems, other simple manipulations can yield similar types of recursive relationships. For example, Richardson derived a deconvolution algorithm by ad hoc manipulations of Bayes rule [12, 13]. Much later it was shown that that Richardson-Lucy algorithm is equivalent to the emission E-ML-EM algorithm discussed in §17.5; by that time convergence of ML-EM had been established, thereby retrospectively confirming convergence of the (equivalent) Richardson-Lucy algorithm. But rarely does a fixed-point story have such a happy ending; usually fixed-point iterations that are based on ad hoc choices for \( M(x) \) can diverge and should be avoided.

11.3 Preconditioned gradient descent (PGD) algorithms (s,opt,pgd)

Choosing \( M(x) \) in (11.2.2) simply to be a fixed positive scalar \( \alpha \), called a step size, multiplying a fixed preconditioning matrix \( P \) yields the following preconditioned gradient descent (PGD) algorithm:

\[
x^{(n+1)} = x^{(n)} - \alpha P \nabla \Psi(x^{(n)}). \tag{11.3.1}
\]

For a general cost function \( \Psi \), one may have difficulty selecting \( P \) to ensure global convergence of \( \{x^{(n)}\} \) or even monotonicity of \( \{\Psi(x^{(n)})\} \). However, for the specific cost functions \( \Psi \) of interest in statistical image reconstruction, we will see numerous subsequent examples (e.g., (12.4)) of convergent algorithms of the form (11.3.1).

Choosing the step size \( \alpha \) properly is quite important. It is rarely the case that \( \alpha = 1 \) will work! Indeed, \( \alpha \) has physical units; in the usual case where \( \Psi \) is unitless and \( P = I \), the units of \( \alpha \) are the square of the units of \( x \), because \( \nabla \Psi \) has units that are the reciprocal of the units of \( x \). The strange units for \( \alpha \) are part of the reason why choosing \( \alpha \) “by hand” is difficult.

11.3.1 Monotonicity conditions for PGD

By assuming properties of \( \Psi \) beyond (11.1.1)-(11.1.2), one can specify sufficient conditions on \( \alpha \) and \( P \) that ensure monotonicity of PGD.

Theorem 11.3.1 If the gradient of \( \Psi \) is \( S \)-Lipschitz continuous per Definition 27.9.16, i.e.,

\[
\|S^{-1}(\nabla \Psi(x) - \nabla \Psi(z))\|_2 \leq \|S'(x - z)\|_2, \quad \forall x, z \in \mathbb{R}^{n_p},
\]

(11.3.2)

where \( S \) is an invertible matrix, \( 0 < \alpha \), and

\[
\alpha P'SS'P < P + P',
\]

(11.3.3)

then the PGD algorithm (11.3.1) monotonically decreases \( \Psi \) and in fact is strictly monotone.

\(^4\) An algorithm is called globally convergent if \( x^{(n)} \to \hat{x} \) for any starting point \( x^{(0)} \).

\(^5\) An algorithm is called locally convergent if \( x^{(n)} \to \hat{x} \) for some nonempty set of initial guesses \( x^{(0)} \) that are “sufficiently close” to \( \hat{x} \).
Proof (extended from [2, p. 21]):
By the first-order Taylor series with remainder (27.8.3) on $\mathbb{R}^n$ and (28.2.8) on $\mathbb{C}^n$:

$$
\Psi(x^{(n)} + z) = \Psi(x^{(n)}) + \nabla \Psi(x^{(n)}) \cdot z + \frac{1}{2} \langle \nabla^2 \Psi(x^{(n)}) z, z \rangle.
$$

Identifying $z = -\alpha Pg$ for $\alpha \in \mathbb{R}$ where $g \equiv \nabla \Psi(x^{(n)})$, we have from (11.3.1), (11.3.2), and the Cauchy-Schwarz inequality (26.4.2):

$$
\Psi(x^{(n)}) - \Psi(x^{(n+1)}) = \nabla \Psi(x^{(n)}) \cdot (-\alpha Pg) + \frac{\alpha}{2} \left\| \nabla \Psi(x^{(n)}) + \alpha \nabla^2 \Psi \right\|_2^2 \geq \frac{\alpha}{2} \left\| \nabla \Psi(x^{(n)}) \right\|_2^2 \geq \frac{\alpha}{2} \left\| \nabla \Psi(x^{(n)}) \right\|_2^2 g \geq 0,
$$

where the last inequality follows from (11.3.3) and §26.2.

When $x^{(n)}$ is not a minimizer, $g \neq 0$, so the final inequality becomes $> 0$, showing that PGD is strictly monotone (see Definition 11.1.1) under the conditions of this theorem.

Corollary 11.3.2 Under Assumption 2 (11.1.2), that $\Psi$ has a finite minimum, and the conditions of Theorem 11.3.1, the gradient $\nabla \Psi(x^{(n)})$ converges to zero as $n \to \infty$ [2, p. 22].

(The proof follows from the inequality $\Psi(x^{(n)}) - \Psi(x^{(n+1)}) \geq \frac{\alpha}{2} \left\| \nabla \Psi(x^{(n)}) \right\|_2^2$ in the proof of Theorem 11.3.1.)

Preconditioners are often assumed to be (Hermitian) symmetric, but $P$ in (11.3.3) need not be symmetric. It does follow from (11.3.3) that $P$ is positive definite in the restricted sense that $x' P x > 0$ for all $x \neq 0$ in $\mathbb{R}^n$. However, if $P$ is not symmetric, the other properties in §26.2 need not hold.

Corollary 11.3.3 If $P = I$, and $S = \sqrt{L} I$, then the Lipschitz condition (11.3.2) simplifies to

$$
\left\| \nabla \Psi(x) - \nabla \Psi(z) \right\|_2 \leq L \left\| x - z \right\|_2, \quad \forall x, z \in \mathbb{R}^n,
$$

where $L$ is called the Lipschitz constant, and the monotonicity condition (11.3.3) simplifies to the following “classical” condition on the step size (cf. (14.5.19)):

$$
0 < \alpha < \frac{2}{L}.
$$

Establishing the Lipschitz condition (11.3.2) or (11.3.4) is perhaps easiest for twice differentiable cost functions; Theorem 27.9.17 shows that if $\Psi$ is twice differentiable, then (11.3.2) holds iff $\left\| \nabla^2 \Psi(x) S^{-1} \right\|_2 \leq 1, \forall x \in \mathbb{R}^n$. If in addition $\Psi$ is convex, then (11.3.2) holds iff $\nabla^2 \Psi(x) \preceq SS'$ by Corollary 27.9.19. In particular, if $\Psi$ is convex and quadratic with Hessian $H$, then (11.3.2) holds with $S = H^{1/2}$. Furthermore, in that convex and quadratic case, if $P$ is symmetric positive definite, then (11.3.3) simplifies to

$$
0 < \alpha < \frac{2}{\lambda_{\text{max}}(PH)}.
$$

Corollary 11.3.4 If $\nabla^2 \Psi(x)$ is bounded by some finite constant $L$, i.e., if the cost function has bounded curvature, then (11.3.2) holds with $P = I$, $S = \sqrt{L} I$, and the classic step size condition (11.3.5).

Example 11.3.5 The cost function $\Psi(x) = \sin(x)$ has second derivative $\hat{\Psi}(x) = \sin(x)$ for which $|\hat{\Psi}(x)| \leq 1$, so $\Psi$ is Lipschitz with $L = 1$. Thus PGD with $P = I$ and $0 < \alpha < 2$ will decrease $\Psi$ monotonically, and $\hat{\Psi}$ will approach zero. But because $\Psi$ is non-convex, PGD will descend towards a local minimizer.

Example 11.3.6 The Huber function $\psi(z)$ in (11.10.9) has derivative $\hat{\psi}(z) = \begin{cases} \frac{t}{\delta}, & |z/\delta| \leq 1 \\ \delta \text{sgn}(z), & \text{otherwise} \end{cases}$. It is not twice differentiable, so we cannot apply Corollary 11.3.4, but one can show (by enumerating a few cases) that $\Psi$ is Lipschitz with $L = 1$. This example illustrates that the Lipschitz gradient condition Theorem 11.3.1 is more general than curvature bounds like Theorem 27.9.17.
11.3.2 Convergence in norm

Theorem 11.3.1 provides sufficient conditions for PGD to decrease the cost function monotonically. We are also interested in examining whether the iterates \{x(n)\} approach a minimizer \(\hat{x}\) monotonically. To provide sufficient conditions for such monotone convergence in norm we focus on cases where \(\Psi\) is twice differentiable with positive definite Hessian \(0 \prec \nabla^2 \Psi\), and hence is strictly convex with unique minimizer \(\hat{x}\) and where \(P\) is also positive definite (and thus has an invertible square root, per Definition 26.2.1). Because \(\Psi(\hat{x}) = 0\) we use (27.8.4) to rewrite the PGD iteration (11.3.1) as

\[
\begin{align*}
x^{(n+1)} - \hat{x} &= x^{(n)} - \hat{x} - \alpha P \left( \nabla \Psi(x^{(n)}) - \nabla \Psi(\hat{x}) \right) \\
&= x^{(n)} - \hat{x} - \alpha P \left( \int_0^1 \nabla^2 \Psi(\hat{x} + \tau(x^{(n)} - \hat{x})) \, d\tau \right) (x^{(n)} - \hat{x}) \\
&= P^{1/2} \left( I - \alpha P^{1/2} \left( \int_0^1 \nabla^2 \Psi(\hat{x} + \tau(x^{(n)} - \hat{x})) \, d\tau \right) P^{1/2} \right) P^{-1/2} (x^{(n)} - \hat{x}).
\end{align*}
\]

Thus

\[
\|P^{-1/2} (x^{(n+1)} - \hat{x})\| \leq \|I - \alpha P^{1/2} \left( \int_0^1 \nabla^2 \Psi(\hat{x} + \tau(x^{(n)} - \hat{x})) \, d\tau \right) P^{1/2} \| \|P^{-1/2} (x^{(n)} - \hat{x})\|, \tag{11.3.10}
\]

and using Lemma 26.2.3, if \(\alpha \nabla^2 \Psi(x) \prec P^{-1}, \forall x\), then \(\|P^{-1/2} (x^{(n)} - \hat{x})\|\), the weighted distance to minimizer \(\hat{x}\), decreases every iteration. If \(P = I\) and \(\|\nabla^2 \Psi\| \leq \mathcal{L}\), then the sufficient condition is \(0 < \alpha < 1/\mathcal{L}\), which is more restrictive than (11.3.5). The following theorem is more general, not requiring strict convexity.

**Theorem 11.3.7** If \(\Psi\) is convex and its gradient \(\nabla \Psi\) is \(\mathcal{S}\)-Lipschitz per (11.3.2), if the set of minimizers \(X^* = \{x^* \in \mathbb{C}^n : \Psi(x^*) \leq \Psi(x), \forall x \in \mathbb{C}^n\}\) is nonempty, and if \(P = [TT']^{-1}\) where \(0 \prec \alpha SS' \prec 2TT'\), then \(\|T'(x^{(n)} - \hat{x})\|\) is monotone nonincreasing for the PGD iteration and \(\{x^{(n)}\}\) converges to some \(x^*\). See Problem 11.4.

Proof: Follow the proof of Theorem 11.10.1.

**Example 11.3.8** Consider the 1D shrinkage problem with \(\Psi(x) = \frac{1}{2} |y - x|^2 + \beta \psi(x)\) where \(0 \leq \psi \leq 1\). Because of this bounded curvature, it is natural to use PGD with \(\alpha P = \frac{1}{\beta + \beta} I\). The error \(x^{(n)} - \hat{x}\) decreases by at least \(\frac{\beta}{\beta + \beta}\) each iteration. If we initialize with \(x^{(0)} = y\) then \(|x^{(0)} - \hat{x}| \leq |y|\) so \(|x^{(n)} - \hat{x}| \leq \left(\frac{\beta}{\beta + \beta}\right)^n |y|\), which can provide a (probably loose) bound on the number of iterations needed to ensure a given error tolerance.

11.3.3 Local convergence rate (s.opt.pgdrate)

An algorithm designer must choose the preconditioner \(P\), and this choice is aided by analysis of the asymptotic convergence rate of the PGD method. If \(\Psi\) is twice differentiable in the neighborhood of a local minimizer \(\hat{x}\), with Hessian \(\hat{H} \triangleq \nabla^2 \Psi(\hat{x})\), then Ostrowski’s theorem [14, p. 300], see (27.14.5), states that the root convergence factor for sequences converging to \(\hat{x}\) is

\[
R_k = \rho(I - \alpha \hat{H}).
\]

To elaborate this rate, we assume that \(\Psi\) is locally twice differentiable i.e., for \(x \approx \hat{x}\):

\[
\Psi(x) \approx \hat{\Psi}(x) \triangleq \hat{\Psi}(\hat{x}) + \frac{1}{2} (x - \hat{x})' \hat{H} (x - \hat{x}),
\]

using (11.1.5). Using this quadratic approximation for \(x^{(n)}\) near \(\hat{x}\), the PGD iterates (11.3.1) are approximately:

\[
x^{(n+1)} \approx x^{(n)} - \alpha P \hat{\Psi}(x^{(n)}) = x^{(n)} - \alpha P \hat{H} (x^{(n)} - \hat{x}).
\]

Assuming \(P\) has an invertible square root (e.g., is positive definite), the residual vector evolves according to the recursion:

\[
P^{-1/2} (x^{(n+1)} - \hat{x}) \approx \left( I - \alpha P^{1/2} \hat{H} P^{1/2} \right) P^{-1/2} (x^{(n)} - \hat{x}).
\]

Hence, to within that approximation:

\[
\|P^{-1/2} (x^{(n+k)} - \hat{x})\| \leq \left\| I - \alpha P^{1/2} \hat{H} P^{1/2} \right\|_2 \|P^{-1/2} (x^{(n)} - \hat{x})\|. \tag{11.3.13}
\]

(Example 11.3.9 below shows that inequality (11.3.13) is tight.) Thus the asymptotic convergence rate of PGD (cf. §27.14) is governed by the spectral radius \(\rho(I - \alpha P^{1/2} \hat{H} P^{1/2})\). Qualitatively speaking, the closer \(\alpha\) is to \(\hat{H}^{-1}\), the faster the convergence asymptotically. For a given \(P\), the best (asymptotic) step size \(\alpha\) is

\[
\alpha_* = \frac{2}{\lambda_{\min}(\hat{H}) + \lambda_{\max}(\hat{H})}, \tag{11.14}
\]
The inequality (11.3.13) is tight. Consider the (separable!) cost function

\[ \text{Example 11.3.9} \]

where

\[ P_{\text{local}} \]

The 11.3.4 Convergence rate of cost function decrease: \( O(1/n) \)

The local convergence rate in \S 11.3.3 characterizes how the iterates \( x^{(n)} \) approach \( \hat{x} \) asymptotically, but says nothing about the early iterations (except when \( \Psi \) is quadratic, in which case (11.3.11) is exact). The following classic theorem bounds the convergence behavior of \( \Psi \) for all iterations.

\[ \text{Theorem 11.3.10} \]

Suppose \( \Psi \) is convex and differentiable, with gradient satisfying a Lipschitz condition of the form (11.3.2), and has a minimizer \( \hat{x} \). If we choose \( \alpha P = [SS']^{-1} \), then the PGD algorithm (11.3.1) produces a sequence \( \{x^{(n)}\} \) for which [16, Thm 3.1] [15]

\[ \Psi(x^{(n)}) - \Psi(\hat{x}) \leq \frac{\|S'(x^{(0)} - \hat{x})\|^2}{2n}, \quad n \geq 1. \]  

(11.3.16)

In the literature this result is often quoted as the “rate of convergence” of PGD is \( O(1/n) \). One must bear in mind that this rate of convergence is for \( \{ \Psi(x^{(n)}) \} \), not \( \{ \|x^{(n)} - \hat{x}\| \} \).

The proof (see Problem 11.6) relies on the following Lemma that generalizes [16, Lemma 2.3].

\[ \text{Lemma 11.11} \]

(See Problem 11.5.) If \( \Psi \) is convex and differentiable, with gradient satisfying a Lipschitz condition of the form (11.3.2), and if we define the PGD update by

\[ M(x) = x - \alpha P \nabla \Psi(x), \]  

(11.3.17)

where \( P \) satisfies (11.3.3), then

\[ \Psi(x) - \Psi(M(z)) \geq \frac{1}{2} \left\| B^{1/2} (M(z) - z) \right\|^2 + \frac{1}{\alpha} \langle z - x, P^{-1}(M(z) - z) \rangle, \]

where by (11.3.3)

\[ B \triangleq \frac{1}{\alpha} (P^{-1} + P^{-T}) - SS' \succeq 0. \]  

(11.3.18)
Generalizing (11.3.16) to consider other preconditioners is an open problem.

The $O(1/n)$ bound in (11.3.16) can be quite pessimistic. For any convex quadratic cost function with Hessian $H$, by (11.3.11)
\[
\Psi(x^{(n)}) - \Psi(\hat{x}) = \frac{1}{2}(x^{(n)} - \hat{x})'H(x^{(n)} - \hat{x}) = \frac{1}{2} \left( P^{-1/2}(x^{(n)} - \hat{x}) \right)'P^{1/2}H P^{1/2} \left( P^{-1/2}(x^{(n)} - \hat{x}) \right)
\]
\[
\leq \frac{1}{2} \left\| P^{1/2}H P^{1/2} \right\| \left\| P^{-1/2}(x^{(n)} - \hat{x}) \right\|^2 \leq \frac{1}{2} \left\| P^{1/2}H P^{1/2} \right\| I - \alpha P^{1/2}H P^{1/2} \left\| P^{-1/2}(x^{(0)} - \hat{x}) \right\|^2,
\]
using (11.3.10). So if $0 < \alpha H < P^{-1}$ then $\Psi$ decreases geometrically, i.e., $O(\rho^n)$, where $\rho = \left\| I - \alpha P^{1/2}H P^{1/2} \right\|$, rather than at the “sublinear” rate of only $O(1/n)$. More generally, for any strongly convex cost function $\Psi$ one can establish a geometric convergence rate for $\|x^{(n)} - \hat{x}\|$ [15, p. 18].

Drori and Teboulle [17] derived a bound like (11.3.16) except with $4n + 2$ in the denominator. They also construct a Huber-like function $\Psi$ for which GD achieves that bound. Thus, the $O(1/n)$ rate is tight over the family of cost functions with Lipschitz gradients.

### 11.3.5 Relationship with optimization transfer (s.opt.pgd.ox)

If $\nabla \Psi$ satisfies the Lipschitz condition (11.3.2), then $\Psi$ has the following quadratic majorizer (cf. Chapter 12):
\[
\Psi(x) \leq \phi^{(n)}(x) \triangleq \Psi(x^{(n)}) + \text{real} \left\{ \nabla \Psi(x^{(n)}), x - x^{(n)} \right\} + \frac{1}{2} \| SS' (x - x^{(n)}) \|^2.
\]

If we choose $P = [SS']^{-1}$ then (11.3.3) simplifies to $0 < \alpha < 2$. In particular, for $\alpha = 1$ the update becomes
\[
x^{(n+1)} = x^{(n)} - [SS']^{-1} \nabla \Psi(x^{(n)}) = \arg \min_{\alpha} \phi^{(n)}(x).
\]

Optimization transfer methods in Chapter 12 typically use $\alpha = 1$, which may not provide the fastest convergence rate in light of (11.3.14). The case analyzed in Theorem 11.3.10 corresponds to $\alpha = 1$ and $\alpha P = [SS']^{-1}$.

### 11.3.6 Step-halving or backtracking (s.opt.pgd.half)

The PGD algorithm in general does not ensure that $\Psi(x)$ decreases monotonically, and often it is challenging to find an appropriate step size $\alpha$. However, one can modify the algorithm by incorporating backtracking [18, p. 131] to ensure descent. Define the search direction
\[
d^{(n)} = -P \nabla \Psi(x^{(n)}),
\]
and replace $\alpha d^{(n)}$ in (11.3.1) with $\alpha_n d^{(n)}$ where we choose $\alpha_n$ to ensure that
\[
\Psi(x^{(n)} + \alpha_n d^{(n)}) \leq \Psi(x^{(n)}).
\]

In particular, one can apply a simple step-halving procedure, where one starts with some initial $\alpha_n$ value and then decreases $\alpha_n$ by a factor of 2 until (11.3.21) is satisfied. Following [18, p. 131], if $P$ is positive definite, then for $d = -P \nabla \Psi(x) \neq 0$, a Taylor series 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 positive definite $P$, and $o(\alpha)/\alpha$ approaches zero as $\alpha \to 0$. Thus, (11.3.20) is a descent direction at $x^{(n)}$, so step-halving is always guaranteed to lead (eventually) to a value $x^{(n+1)}$ that decreases $\Psi$. However, obtaining this guarantee incurs the computational price of multiple evaluations of $\Psi(x^{(n)} + \alpha d^{(n)})$.

A problem with step-halving is that it is possible that the step size could become smaller each iteration possibly preventing complete convergence. A solution to this problem is to apply the Armijo rule [3, p 29], a modified line search that ensures a sufficient decrease in the cost function each iteration.

We call these types of approaches forced monotonic methods. In contrast, several of the algorithms described in this book are intrinsically monotonic, and guarantee decreases in $\Psi$ without any line search or backtracking.

### 11.3.7 Ideal preconditioner (s.opt.precon)

From the analysis in §11.3.3, for quadratic cost functions, the ideal preconditioner would be $P_0 = H^{-1}$ so that $P_0 H = I$, because the $n_p \times n_p$ identity matrix $I$ has the minimal condition number (unity), and the PSD algorithm would converge in one step. For nonquadratic $\Psi$, the inverse-Hessian preconditioner $P_0(x) = H^{-1}(x)$ would yield superlinear convergence rates akin to the Newton-Raphson method [19]. Because we cannot compute $H^{-1}$ for large $n_p$, one must develop preconditioners that approximate $H^{-1}$; see Chapter 19.
11.3.8 Preconditioning as a coordinate transformation

If \( P = TT' \) where \( T \) is invertible and we consider the change of variables \( z = T^{-1}x \), then we can express the (unconstrained) minimization problem (11.1.3) as follows:

\[
\hat{z} = \arg \min_z \Psi_P(z), \quad \Psi_P(z) \triangleq \Psi(Tz).
\]  
(11.3.22)

A gradient descent algorithm for \( \Psi_P \) in terms of \( z \) has the form

\[
z^{(n+1)} = z^{(n)} - \alpha \nabla \Psi_P(z^{(n)}) = z^{(n)} - \alpha T' \nabla \Psi(Tz^{(n)}),
\]

by applying the chain rule. Multiplying by \( T \) yields the PGD update (11.3.1). So preconditioning is equivalent to a coordinate transformation. Geometrically, a good preconditioner is related to coordinates in which the cost function has nearly spherical level sets, because in such a coordinate system the gradient vector points towards the minimizer.

11.4 Newton-Raphson algorithm (s,opt,nr)

From the preceding convergence analysis, the optimal preconditioner would be \( \alpha P = H(\hat{x})^{-1} \). Because \( \hat{x} \) is unknown, this preconditioner is impractical. However, in principle we can achieve comparable performance by using the current Hessian \( H_n = H(x^{(n)}) \) instead of \( H(\hat{x}) \). This leads to the Newton-Raphson algorithm, often called simply Newton’s method, which we develop in this section using a slightly different approach.

If \( \Psi \) is twice differentiable, then we can make a 2nd-order Taylor series approximation\(^6\) using (11.1.8):

\[
\Psi(x) \approx \Psi_n(x) \triangleq \Psi(x^{(n)}) + \text{real}\{\langle \nabla \Psi(x^{(n)}), x - x^{(n)} \rangle\} + \frac{1}{2} (x - x^{(n)})' H_n (x - x^{(n)}).
\]  
(11.4.1)

Equating to zero the gradient of the 2nd-order approximation \( \Psi_n \) yields the following necessary condition for a minimizer of that approximation:

\[
0 = \nabla \Psi_n(x) \bigg|_{x = x^{(n+1)}} = \nabla \Psi(x^{(n)}) + H_n (x - x^{(n)}) \bigg|_{x = x^{(n+1)}}.
\]  
(11.4.2)

Solving for \( x^{(n+1)} \) yields the following classical Newton-Raphson algorithm:

\[
x^{(n+1)} = x^{(n)} - \left[ \nabla^2 \Psi(x^{(n)}) \right]^{-1} \nabla \Psi(x^{(n)}),
\]  
(11.4.3)

Typically this algorithm is entirely impractical for imaging problems, due to the size of the Hessian. In addition, it is not guaranteed to monotonically decrease \( \Psi \). (One can attempt to overcome this limitation using backtracking, see (11.3.21), at least for convex \( \Psi \), but in general adding a line search does not ensure convergence [20].) When \( \Psi \) is strongly convex, and when \( \nabla^2 \Psi \) satisfies a Lipschitz condition, one can show that Newton-Raphson is locally convergent with a quadratic convergence rate: \( \|x^{(n)} - \hat{x}\| \leq c \|x^{(0)} - \hat{x}\|^q \), for a constant \( c \) that depends on \( \Psi \) and a constant \( q < 1 \) that depends on \( \Psi \) and \( x^{(0)} \) [15, p. 15] [2, p. 28]. This very desirable fast convergence property is shared by almost none of the (practical) algorithms discussed in this book.

Variations of the Newton-Raphson algorithm for nonlinear least-squares problems include the Gauss-Newton and Levenberg-Marquardt methods; see §12.13.

11.5 Preconditioned steepest descent (PSD) algorithms (s,opt,psd)

In the PGD algorithm described in (11.3.1) above, the “step size” remains constant each iteration. Often it is impractical to determine the best step size (11.3.14), both because the eigenvalues are unknown and because the Hessian \( \hat{H} \) depends on the unknown minimizer \( \hat{x} \). To circumvent these difficulties, one can let the preconditioned gradient vector define a search direction and then seek the minimizer of \( \Psi(\cdot) \) along that direction. With the search direction \( d^{(n)} \) defined as the (negative) preconditioned gradient in (11.3.20), the preconditioned steepest descent (PSD) algorithm [21] is given as follows:

\[
d^{(n)} \triangleq -P \nabla \Psi(x^{(n)})
\]

\[
\alpha_n \triangleq \arg \min_{\alpha \in [0,\infty)} \Psi(x^{(n)} + \alpha d^{(n)})
\]  
(11.5.1)

\[
x^{(n+1)} = x^{(n)} + \alpha_n d^{(n)},
\]  
(11.5.2)

---

\(^6\) Readers should compare (11.4.1), which is useful for algorithm design, to (11.3.11), which is useful for algorithm analysis. See also (28.4.2).
where $\alpha_n$ is called the **step size** or **step length**. The one-dimensional minimization in (11.5.1) is called a **line search**. For a general nonquadratic cost function $\Psi$, this search may add considerable computational expense per iteration. Often one must choose between using PGD with a conservative step size (thus requiring more iterations) and PSD that needs more work per iteration but requires fewer iterations.

One could argue that (11.5.2) is not really an “algorithm” because the method for the required minimization (11.5.1) remains unspecified. See §11.6 for discussion of line search methods.

### 11.5.1 Orthogonality and search directions

A necessary condition for finding the minimizing step size $\alpha_n \in [0, \infty)$ in (11.5.1) is that

$$0 = \frac{\partial}{\partial \alpha} \Psi(x^{(n)} + \alpha d^{(n)}) \bigg|_{\alpha = \alpha_n} = \langle \nabla \Psi(x^{(n)} + \alpha d^{(n)}), d^{(n)} \rangle \bigg|_{\alpha = \alpha_n} = \langle \nabla \Psi(x^{(n+1)}), d^{(n)} \rangle,$$

by applying the chain rule and (11.5.2). In other words, the next gradient $\hat{\nabla} \Psi(x^{(n+1)})$ and the current search direction $d^{(n)}$ are orthogonal, at least if the line search finds the exact minimizer. This property is illustrated in Fig. 11.3.1. For an **inexact line search** that finds only an approximate minimizer in (11.5.1), the next gradient $\hat{\nabla} \Psi(x^{(n+1)})$ is approximately orthogonal to $d^{(n)}$.

### 11.5.2 Complex case

We can also apply PSD to cost functions with complex-valued arguments, *i.e.*, $\Psi : \mathbb{C}^n \to \mathbb{R}$, where $x^{(n)}$, $d^{(n)} \in \mathbb{C}^n$, provided we define $\nabla \Psi$ appropriately; see (28.2.6) in Appendix 28. The line search (11.5.1) is still over $\alpha \in [0, \infty)$, even when $x$ is complex. For complex cases,

$$\frac{\partial}{\partial \alpha} \Psi(x^{(n)} + \alpha d^{(n)}) = \text{real}\{\langle \nabla \Psi(x^{(n)} + \alpha d^{(n)}), d^{(n)} \rangle\}$$

and the orthogonality condition (11.5.3) becomes:

$$\text{real}\{\langle \nabla \Psi(x^{(n+1)}), d^{(n)} \rangle\} = 0.$$

See pgd_step.m.

### 11.5.3 Asymptotic convergence rate

Choosing the preconditioner $P$ follows similar analysis as in §11.3.3. Using the quadratic approximation (11.3.11), for $x^{(n)}$ near $\hat{x}$, the PSD iterates are approximately

$$x^{(n+1)} = x^{(n)} + \hat{\alpha}_n \hat{d}^{(n)},$$

where $\hat{d}^{(n)} = -P \nabla \Psi(x^{(n)}) = -P \hat{H}(x^{(n)} - \hat{x})$ and

$$\hat{\alpha}_n \triangleq \arg \min_{\alpha} \hat{\Psi}\left(x^{(n)} + \alpha \hat{d}^{(n)}\right) = \frac{\langle \hat{x} - x^{(n)}, \hat{H} \hat{d}^{(n)} \rangle}{\langle \hat{d}^{(n)}, \hat{d}^{(n)} \rangle}.$$

By this construction, it follows that

$$\left\|x^{(n+1)} - \hat{x}\right\|_{\hat{H}^{1/2}} = \min_{\alpha} \left\|x^{(n)} + \alpha \hat{d}^{(n)} - \hat{x}\right\|_{\hat{H}^{1/2}} = \min_{\alpha} \left\|I - \alpha P \hat{H}\right\| \left(x^{(n)} - \hat{x}\right)\right\|_{\hat{H}^{1/2}}.$$ 

Assuming $P$ and $\hat{H}$ are positive definite and defining the (weighted) error vector $\delta^{(n)} = P^{-1/2}(x^{(n)} - \hat{x})$ and the preconditioned Hessian matrix $\hat{H} = P^{1/2} \hat{H} P^{1/2}$, it follows from (11.5.5) that

$$\left\|\delta^{(n+1)}\right\|_{\hat{H}^{1/2}} = \min_{\alpha} \left\|I - \alpha \hat{H}\right\| \delta^{(n)}\right\|_{\hat{H}^{1/2}} \leq \min_{\alpha} \left\|I - \alpha \hat{H}\right\| \right\|\delta^{(n)}\|_{\hat{H}^{1/2}}.$$

In particular, following [22, p. 31],

$$\left\|I - \alpha \hat{H}\right\|_{\hat{H}^{1/2}} = \arg \max_{\alpha \neq 0} \sqrt{\frac{x^T (I - \alpha \hat{H}) \hat{H} (I - \alpha \hat{H}) x}{x^2}} = \arg \max_{j=1, \ldots, n_p} |1 - \alpha \lambda_j| \sqrt{\lambda_j},$$

where the $\lambda_j$ values are the eigenvalues of $\hat{H}$. Thus

$$\min_{\alpha} \left\|I - \alpha \hat{H}\right\|_{\hat{H}^{1/2}} = \min_{\alpha} \arg \max_{j=1, \ldots, n_p} |1 - \alpha \lambda_j| \sqrt{\lambda_j} = \frac{k - 1}{k + 1},$$

(11.5.6)
where \( \kappa \triangleq \lambda_{\text{max}}(\tilde{H})/\lambda_{\text{min}}(\tilde{H}) \) is the condition number of \( \tilde{H} \). Note that the minimizing \( \alpha \) for (11.5.7) is \( \alpha_n \), whereas that for the upper bound (11.5.6) is \( \alpha = \frac{\lambda_{\text{min}}(\tilde{H}) + \lambda_{\text{max}}(\tilde{H})}{2} \). Combining the above relationships, the weighted error norm decreases each iteration at least as much as the following [1, p. 32]:

\[
\| \delta^{(n+1)} \|_{\tilde{H}^{1/2}} \leq \frac{\kappa - 1}{\kappa + 1} \| \delta^{(n)} \|_{\tilde{H}^{1/2}}. \tag{11.5.7}
\]

So the closer \( \kappa \) is to unity, the faster the convergence should be. By (26.1.1): \( \kappa = \kappa(\tilde{H}) = \kappa(P^{1/2} \tilde{H} P^{1/2}) = \kappa(P \tilde{H}) \), so one would like to choose \( P \) to minimize the condition number of the product \( P \tilde{H} \).

It is interesting that (11.5.7) for PSD is similar to (11.3.15) PGD yet PSD does not require determining \( \alpha_* \).

Example 11.3.9 shows that the inequality (11.5.7) is tight. (In that example, the line search yields \( \alpha_n = \alpha_* \) every iteration.)

### 11.6 Line search methods (s,opt,line)

Several of the algorithms described in this chapter, including PSD and PCG, require a line search like (11.5.1). There are many “classical” methods for solving this 1D minimization problem, such as the golden section search [wiki] and the bisection method (or binary search) [wiki]. See also [26] [27]. A monotonic surrogate-function method well-suited to inverse problems is described in §12.5.6. Rarely is the minimization in (11.5.1) exact. Often one seeks \( \alpha_n \) that satisfies the Wolfe conditions [28, 29] [wiki], including the Armijo rule [30]:

\[
\Psi(x^{(n)} + \alpha_n d^{(n)}) \leq \Psi(x^{(n)}) + c_1 \alpha_n \text{real}\{\nabla \Psi(x^{(n)}), \ d^{(n)}\}
\]

where \( 0 < c_1 \ll 1 \).

Most conventional methods need some initial guess for the step size. This section summarizes some alternate methods that consider the properties of the cost function \( \Psi \) to help accelerate the line search.

#### 11.6.1 Line search using Lipschitz conditions (s,opt,line,lips)

If the gradient \( \nabla \Psi \) of the cost function is \( S \)-Lipschitz continuous per (11.3.2), then the problem of minimizing the 1D function \( f(\alpha) = \Psi(x + \alpha d) \) is simplified because \( f \) itself also has a Lipschitz continuous derivative:

\[
|f(\alpha) - f(b)| = |\text{real}\{\nabla \Psi(x + \alpha d) - \nabla \Psi(x + b d), \ d\}| \\
\leq \| S^{-1}(\nabla \Psi(x + \alpha d) - \nabla \Psi(x + b d)) \cdot S'd \| \\
\leq \| S^{-1}(\nabla \Psi(x + \alpha d) - \nabla \Psi(x + b d)) \| \| S'd \| \\
\leq \| S'(\alpha d - b d) \| \leq \| S'd \|^2 |\alpha - b|.
\]

Thus \( f \) has Lipschitz constant \( L_f = \| S'd \|^2 = d'SS'd \), and we can apply gradient descent to \( f \):

\[
\alpha^{(k+1)} = \alpha^{(k)} - \frac{1}{L_f} f(\alpha^{(k)}) = \alpha^{(k)} - \frac{1}{L_f} \text{real}\{\nabla \Psi(x + \alpha^{(k)} d), \ d\}. 
\]

By Theorem 11.3.1, this simple iteration is guaranteed to decrease \( f(\alpha) \) monotonically. For an example, see Problem 11.7.

#### 11.6.2 Step size using Newton’s method (s,opt,line,newt)

For cost functions that are twice differentiable and approximately quadratic, one can use the 1D version of Newton’s method as a reasonable starting point for choosing the step size \( \alpha_n \in [0, \infty) \). Define

\[
\psi(\alpha) = \Psi(x^{(n)} + \alpha d^{(n)}).
\]

Then by (11.5.3) or (11.5.4):

\[
\dot{\psi}(\alpha) = \text{real}\{\nabla \Psi(x^{(n)} + \alpha d^{(n)}), \ d^{(n)}\}
\]

and \(^7\)

\[
\ddot{\psi}(\alpha) = \langle d^{(n)}, \nabla^2 \Psi(x^{(n)} + \alpha d^{(n)}) d^{(n)} \rangle \tag{11.6.1}
\]

\[
\approx \frac{1}{\epsilon} \left( \dot{\psi}(\alpha + \epsilon) - \dot{\psi}(\alpha) \right) \tag{11.6.2}
\]

\(^7\) In complex case, we assume \( \nabla \Psi \) is holomorphic and define \( \nabla^2 \Psi \) as in §28.4.
\[ \alpha_n = -\frac{\psi(0)}{\psi(0)} \approx \frac{1}{\epsilon} \left( \text{real}\{ \nabla \Psi(x^{(n)} + (\alpha + \epsilon)d^{(n)}) \} - \text{real}\{ \nabla \Psi(x^{(n)}), d^{(n)} \} \right). \]  

Thus the approximate minimizer is

\[ \alpha_n = -\frac{\psi(0)}{\psi(0)} \approx \frac{1}{\epsilon} \left( \text{real}\{ \nabla \Psi(x^{(n)}), d^{(n)} \} \right). \]

This approximation requires one extra evaluation of the cost function gradient \( \nabla \Psi \) and avoids computing the Hessian of \( \Psi \). If this step size is too large, then back track as described in §11.3.6.

### 11.6.3 Line search for PWLS cost functions (s,opt,line,pwls)

For general cost functions, repeatedly evaluating \( \Psi \) or its gradient to perform a line search may be expensive. Fortunately, for the PWLS cost functions of interest in many imaging problems, we can evaluate \( \Psi \) and its gradient for a line search fairly efficiently. If

\[ \Psi(x) = \frac{1}{2} \| y - Ax \|^2 + \beta \sum_k \psi([C x]_k) \]

then we can write the line search cost function as

\[ f(\alpha) = \Psi(x^{(n)} + \alpha d^{(n)}) = \frac{1}{2} \| y - A(x^{(n)} + \alpha d^{(n)}) \|^2 + \beta \sum_k \psi([C(x^{(n)} + \alpha d^{(n)})]_k) \]

\[ = \frac{1}{2} \| y - (Ax^{(n)} + \alpha(Ad^{(n)}) \|^2 + \beta \sum_k \psi([C x^{(n)}]_k + \alpha[Cd^{(n)}]_k). \]

We can precompute the products \( Ax^{(n)}, Ad^{(n)}, C x^{(n)} \) and \( Cd^{(n)} \) before starting the line search, and then use (11.6.5). Note that

\[ Ax^{(n+1)} = A(x^{(n)} + \alpha_n d^{(n)}) \]

\[ = (Ax^{(n)}) + \alpha_n(Ad^{(n)}), \]

so we can determine \( Ax^{(n+1)} \) from the vectors \( Ax^{(n)} \) and \( Ad^{(n)} \) computed before starting the line search.

### 11.7 Accelerated first-order methods (s,opt,a1)

The PGD method (11.3.1) is a first-order method because it depends only on the gradient of the cost function and not on its second derivatives (i.e., the Hessian).

The sufficient condition for monotone convergence (11.3.2) also involves "only" \( \nabla \Psi \), although often one determines the Lipschitz constant by finding (an upper bound on) the maximum eigenvalue of the Hessian of \( \Psi \) via Corollary 11.3.4.

First-order methods are appealing for large-scale optimization problems (such as those arising in image reconstruction) because the Hessian is too large to store. However, traditional first-order methods like PGD often converge undesirably slowly. This has led to considerable interest in accelerated first-order methods, some of which are summarized in this section. This field is evolving rapidly [31].

### 11.7.1 Barzilai and Borwein gradient method (s,opt,bb)

The Barzilai and Borwein gradient method (BBGM) for large-scale optimization problems is the following simple iteration [32–35]:

\[
\begin{align*}
\mathbf{g}^{(n)} & \triangleq \nabla \Psi(x^{(n)}) \\
\alpha_n & = \frac{\| x^{(n)} - x^{(n-1)} \|^2}{\mathbf{g}^{(n)} - \mathbf{g}^{(n-1)}_n} \\
x^{(n+1)} & = x^{(n)} - \alpha_n \mathbf{g}^{(n)}.
\end{align*}
\]

This method has global convergence when \( \Psi \) is a strictly convex quadratic, even though it is not guaranteed to decrease the cost function monotonically [33, 36].

For non-quadratic problems, it is important to modify the algorithm to ensure that it tends to descend, but enforcing strict descent would reduce it to the PSD method that usually converges slowly. Raydan [34] proposed a modification that adjusts the step size \( \alpha_n \) to enforce the following condition:

\[ \Psi(x^{(n+1)}) \leq \max_{0 \leq j \leq M} \Psi(x^{(n-j)}) + \gamma (x^{(n+1)} - x^{(n)}, g^{(n)}), \]

where \( M \) is a nonnegative integer and \( \gamma \) is a small positive number, following [37, 38].

Using §11.3.8, one can derive the following preconditioned version. (Problem 11.8.)
Example 11.7.1 Fig. 11.7.1 illustrates BBGM for the same problem described in Example 11.7.1 for two choices of initial step size $\alpha_0$. Clearly the iterates can be non-monotonic.

Figure 11.7.1: Fast convergence of BBGM for the same example shown in Fig. 11.3.1. In this (strongly convex, quadratic) case, BBGM with $\alpha_0 = 1$ (green) converges exactly in 3 iterations, and converges rapidly with $\alpha_0 = 1/2$ (blue).

MIRT See qpwls_bb1.m.

11.7.2 Nesterov’s “optimal” first-order methods

[42] [43] proposed accelerated gradient descent methods that use momentum. His methods were described for cost functions having gradients that satisfy the classical Lipschitz condition (11.3.5). Using the coordinate change ideas of §11.3.8, one can generalize to cases that are $S$-Lipschitz continuous per (11.3.2) (cf. [44]) leading to the following algorithm. Starting with $z^{(0)} = x^{(0)}$ and $t_0 = 1$ Nesterov’s fast gradient method (FGM) is:

$$
t_{n+1} = \frac{1 + \sqrt{1 + 4t_n^2}}{2}
$$

$$
x^{(n+1)} = z^{(n)} - [SS']^{-1} \nabla \Psi(z^{(n)})
$$

$$
= \arg \min_x \phi^{(n)}(x), \quad \phi^{(n)}(x) = \Psi(z^{(n)}) + \nabla \Psi(z^{(n)}) (x - z^{(n)}) + \frac{1}{2} \|x - z^{(n)}\|^2_{SS'}
$$

$$
z^{(n+1)} = x^{(n+1)} + \frac{t_n - 1}{t_{n+1}} (x^{(n+1)} - x^{(n)}).
$$

The ratio $\frac{t_n - 1}{t_{n+1}}$ for the momentum term approaches 2 as $n$ increases. Because $t_0 = 1$, the first iteration is simply PGD with preconditioner $P = [SS']^{-1}$.

This method reaches an “$\epsilon$ optimal solution” where $\Psi(x^{(n)}) \leq \Psi(\hat{x}) + \epsilon$ within $O(1/\sqrt{\epsilon})$ iterations, whereas conventional PGD requires $O(1/\epsilon)$ iterations [42] [43]. Specifically, if $\Psi$ is convex and has Lipschitz gradient, then

$$
\Psi(x^{(n)}) - \Psi(\hat{x}) \leq \frac{2 \|S'(x^{(0)} - \hat{x})\|^2}{n^2}.
$$

(11.7.1)

This $O(1/n^2)$ convergence rate is optimal in a certain (non-asymptotic) sense [4, p. 77]. One can generalize this algorithm and inequality to include diagonal preconditioners [45]. Generalizing to other preconditioners is an open problem. Nesterov’s methods require a Lipschitz constant $L$ or matrix $S$, whereas PSD, BBGM, and PCG do not.
Figure 11.7.2: Illustration of Nesterov method for the same example shown in Fig. 11.3.1. For the blue curve, $L = 2\rho(\nabla^2 \Psi)$, reflecting the fact that often the maximum eigenvalue is over-estimated. For the green curve, $L = \rho(\nabla^2 \Psi)$, which is the ideal Lipschitz constant. In this (strongly convex, quadratic) case, the momentum term causes the iterates to slightly overshoot the minimizer.

See [46] for a version that estimates the Lipschitz constant locally, during the iteration.

Fig. 11.7.2 shows example based on Example 11.3.9, using $S = \sqrt{L}I$. The bound (11.7.1) is tight, to within a constant factor, because there exists a convex function for which FGM converges at a rate within a constant factor of the lower bound [4, Thm 2.1.7] [17].

Recently the bound (11.7.1) has been tightened by a factor of two using a similarly efficient algorithm [47–49]. And the optimized gradient method (OGM) in [47] is an optimal first-order method [50].

11.8 Preconditioned conjugate gradients (PCG) (s,opt,pcg)

Both the PGD and PSD algorithms above choose the search direction according to the preconditioned gradient vector (11.3.20). Example 11.3.9 illustrates that (11.3.20) is a somewhat inefficient choice of search direction, because PSD required multiple iterations even for a separable 2D cost function. This section describes an improvement called the preconditioned conjugate gradient (PCG) method. It is also called nonlinear CG when the cost function is nonquadratic [wiki].

To further accelerate convergence, conjugate-gradient methods modify the search directions to ensure that they are mutually conjugate (or approximately so for nonquadratic problems), meaning that they are orthogonal with respect to an inner product related to the Hessian of $\Psi$. This approach ensures a more efficient search over the parameter space, and even leads to convergence in $n_p$ iterations [22, p. 36] for quadratic cost functions (in the absence of numerical roundoff errors).

We would like to choose a search direction $d^{(n)}$ that is (approximately) $H_{n-1}$-orthogonal to the previous search direction $d^{(n-1)}$, where $H_n$ is the Hessian of $\Psi$ at $x^{(n)}$. Specifically, we would like to choose $d^{(n)}$ such that

$$\langle d^{(n)}, H_{n-1} d^{(n-1)} \rangle = 0.$$  \hspace{1cm} (11.8.1)

After choosing a suitable search direction, PCG uses the PSD approach of a line search (11.5.1) and the corresponding update (11.5.2).

A simple way to achieve conjugacy (11.8.1) is to design $d^{(n)}$ using the recursion\footnote{Provided that $P$ is a Hermitian positive definite preconditioning matrix, one can motivate (11.8.2) by working in the transformed coordinate system discussed in §11.3.8.}

$$d^{(n)} = -P g^{(n)} + \gamma_n d^{(n-1)},$$  \hspace{1cm} (11.8.2)

where $g^{(n)} \triangleq \nabla \Psi(x^{(n)})$, and to choose $\gamma_n$ to satisfy (11.8.1). Substituting into (11.8.1) and simplifying yields

$$\gamma_n D \triangleq \frac{\langle H_{n-1} d^{(n-1)}, P g^{(n)} \rangle}{\langle H_{n-1} d^{(n-1)}, d^{(n-1)} \rangle}.$$  \hspace{1cm} (11.8.3)

Proposed by Daniel [51], this choice for $\gamma_n$ satisfies the conjugacy condition (11.8.1) exactly, but is inconvenient (for nonquadratic cost functions) because it depends on the Hessian $H_{n-1}$. Furthermore, after multiple iterations of the recursion (11.8.3), it is possible that the search direction $d^{(n)}$ will not be a descent direction. Several alternative choices for $\gamma_n$ have been proposed that overcome these disadvantages. Historically [52], these alternatives were derived by
a variety of methods, often based on generalizing the PCG method for quadratic cost functions. Here we present the alternatives as approximations to (11.8.3).

Using the 2nd-order Taylor series (11.4.1) and its gradient (11.4.2) leads to the following approximation:

\[ g^{(n)} \approx g^{(n-1)} + H_{n-1}(x^{(n)} - x^{(n-1)}) \]  
\[ = g^{(n-1)} + \alpha_{n-1}H_{n-1}d^{(n-1)}, \]  

(11.8.4) (11.8.5)

exploiting (11.5.2), i.e., \( H_{n-1}d^{(n-1)} = (g^{(n)} - g^{(n-1)})/\alpha_{n-1} \). (This relation is exact for quadratic cost functions.) Substituting into (11.8.3) yields the Hestenes-Stiefel formula [53]:

\[ \gamma_n^{HS} \triangleq \frac{(g^{(n)} - g^{(n-1)}, P g^{(n)})}{(g^{(n)} - g^{(n-1)}, d^{(n-1)})}, \]

(11.8.6)

For further variations, invoke the orthogonality condition (11.5.3) or (11.5.4), i.e., \( \alpha = \text{real} \{ (g^{(n)}, d^{(n-1)}) \} \), which holds for an exact line search. Assuming the direction design (11.8.2) is used in the previous iteration too, the denominator in (11.8.6) simplifies as follows:

\[ (g^{(n)} - g^{(n-1)}, d^{(n-1)}) = -(g^{(n-1)}, d^{(n-1)}) = -(g^{(n-1)}, -P g^{(n-1)} + \gamma_n^{HS}d^{(n-2)}) = (g^{(n-1)}, P g^{(n-1)}). \]

Substituting into (11.8.6) yields the Polak-Ribiere [54, 55] choice:

\[ \gamma_n^{PR} \triangleq \frac{(g^{(n)} - g^{(n-1)}, P g^{(n)})}{(g^{(n)} - g^{(n-1)}, d^{(n-1)})} \]

(11.8.7)

This choice has been used frequently for image reconstruction [26, 56, 57].

Turning now to the numerator in (11.8.6), note that by (11.8.2) and (11.8.5):

\[ \langle Pg^{(n-1)}, g^{(n)} \rangle = \langle d^{(n-1)} - \gamma_n^{HS}d^{(n-2)}, g^{(n)} \rangle = -\gamma_n^{HS} \langle d^{(n-2)}, g^{(n)} \rangle \]

\[ \approx -\gamma_n^{HS} \langle d^{(n-2)}, g^{(n-1)} + \alpha_{n-1}H_{n-1}d^{(n-1)} \rangle \]

\[ = \gamma_n^{HS}\alpha_{n-1} \langle d^{(n-2)}, H_{n-1}d^{(n-1)} \rangle \approx 0, \]

(11.8.9) (11.8.10)

where the last approximation follows from the conjugacy condition (11.8.1) provided that \( H_{n-1} \approx H_{n-2} \). Applying (11.8.10) to (11.8.6) yields the formula of Dai and Yuan [58]:

\[ \gamma_n^{DY} \triangleq \frac{(g^{(n)} - g^{(n-1)}, P g^{(n)})}{(g^{(n)} - g^{(n-1)}, d^{(n-1)})}. \]

(11.8.11)

Alternatively, applying (11.8.10) to (11.8.7) yields the classic Fletcher-Reeves formula [59]:

\[ \gamma_n^{FR} \triangleq \frac{(g^{(n)}, P g^{(n)})}{(g^{(n)} - g^{(n-1)}, P g^{(n-1)})}. \]

(11.8.12)

Because \( \alpha = \langle g^{(n)}, d^{(n-1)} \rangle \) for an exact line search, \( \langle P (g^{(n)} - g^{(n-1)}) - \zeta_n d^{(n-1)}, g^{(n)} \rangle \) is an alternative to the numerator of (11.8.6) for any value of \( \zeta_n \). Hager and Zhang [27, 60, 61] surveyed numerous choices for \( \gamma_n \) and proposed the following choice that has favorable convergence properties:

\[ \gamma_n^{HZ} \triangleq \frac{P (g^{(n)} - g^{(n-1)}) - \zeta_n d^{(n-1)}, g^{(n)}}{(g^{(n)} - g^{(n-1)}, d^{(n-1)})}, \]

\[ \zeta_n \triangleq \frac{2(g^{(n)} - g^{(n-1)}, P (g^{(n)} - g^{(n-1)}))}{(g^{(n)} - g^{(n-1)}, d^{(n-1)})}. \]

(11.8.13)

(Problem 11.9.) This choice ensures monotonicity under mild conditions, unlike many of the previous alternatives. See also [19, 62-72].

In practice, often one uses \( [\gamma_n]_+ \) in place of \( \gamma_n \) for inexact line searches [27, 73]. For cases where \( x \) is complex, see [74-76].

The following summarizes the PCG algorithm for one of the many options for \( \gamma_n \).

**PCG Algorithm (Polak-Ribiere version of \( \gamma_n \))**

\[
\begin{align*}
g^{(n)} &= \nabla \Psi(x^{(n)}) \quad &\text{(gradient)} \\
p^{(n)} &= P g^{(n)} \quad &\text{(precondition)} \\
\gamma_n &= \begin{cases} 
0, & n = 0 \\
\text{real} \{ (g^{(n)} - g^{(n-1)}, p^{(n)}) \} / \text{real} \{ (g^{(n-1)}, p^{(n-1)}) \}, & n > 0
\end{cases} \quad &\text{(Polak-Ribiere)} \\
d^{(n)} &= -p^{(n)} + \gamma_n d^{(n-1)} \quad &\text{(search direction)} \quad (11.8.14) \\
\alpha_n &= \arg \min_{\alpha} \Psi(x^{(n)} + \alpha d^{(n)}) \quad &\text{(step size)} \quad (11.8.15) \\
x^{(n+1)} &= x^{(n)} + \alpha_n d^{(n)} \quad &\text{(update)} \quad (11.8.16)
\end{align*}
\]
For nonquadratic cost functions $\Psi$, one must find $\alpha_n$ using a line-search. \S 11.6 and \S 12.5.6 describe efficient line-search methods suitable for many imaging problems, based on \textit{majorize-minimize} methods to determine the step sizes [57] [77].

Generalizations of PCG using multiple search directions and/or preconditioners have also been proposed [78–81]. A related generalization is subspace minimization using multiple search directions simultaneously [82].

### 11.8.1 Asymptotic convergence rate

Analysis of the convergence rate of the PCG algorithm is considerably more complicated than for the steepest descent algorithm. For quadratic cost functions, the errors decrease at least as rapidly as the following bound [22, p. 51]:

$$
\|x^{(n)} - \hat{x}\|_{H^{1/2}} \leq 2 \left( \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^n \|x^{(0)} - \hat{x}\|_{H^{1/2}},
$$

where $\kappa$ is the \textbf{condition number} of the preconditioned Hessian $P^{1/2}H P^{1/2}$ corresponding to the cost function $\Psi(P^{1/2})$. Comparing to the convergence rate of PSD given in (11.5.7), we see that PCG has considerably faster convergence (due to the square root of $\kappa$). Even tighter error bounds as a function of iteration, involving $n$th-order polynomials with roots near the eigenvalues of the (preconditioned) Hessian, are given in [22, p. 51].

Recently, nonlinear PCG algorithms have been developed that are globally convergent under certain conditions on the line search [27, 60, 61].

### 11.9 Quasi-Newton methods (s.opt.qn)

The preconditioning matrix $P$ remains unchanged throughout the iterations of the preceding algorithms, except in the Newton-Raphson iteration (11.4.3), in which $P$ is the inverse Hessian, which is impractical to compute. Quasi-Newton methods attempt to provide a practical approach involving more easily inverted matrices, yet hopefully accelerating convergence (relative to algorithms with a fixed preconditioner) by \textit{updating} the preconditioning matrix each iteration in hopes of forming an improved approximation to the inverse of the Hessian matrix. A basic quasi-Newton method uses the rank-one update of Davidon [83] to adjust the preconditioner each iteration, see [18, p. 136] and [84].

A \textbf{quasi-Newton} algorithm begins with some initial guess $P_0$ of the inverse Hessian $H^{-1}(\hat{x})$, and updates both $x^{(n)}$ and $P_n$ each iteration. The principle behind quasi-Newton methods is the \textbf{secant condition}, which originates from the gradient of the quadratic approximation (11.4.1) evaluated at $x^{(n+1)}$:

$$
\nabla \Psi(x^{(n+1)}) \approx \nabla \Psi(x^{(n)}) + H_n(x^{(n+1)} - x^{(n)}).
$$

Defining

$$
\delta^{(n)} \triangleq x^{(n+1)} - x^{(n)}
$$

and

$$
q^{(n)} \triangleq \nabla \Psi(x^{(n+1)}) - \nabla \Psi(x^{(n)})
$$

we see that $q^{(n)} \approx H_n \delta^{(n)}$. The (inverse) secant condition, which constrains the update of $P_n$, is defined as follows:

$$
P_{n+1} q^{(n)} = \delta^{(n)}.
$$

Clearly there are many matrices that satisfy this condition. For simplicity, Davidon constrained $P_n$ to evolve via a \textbf{rank-one update} (so called because the matrix $w(n)[w(n)]'$ has unity rank) of the following form:

$$
P_{n+1} = P_n + b_n w^{(n)} [w^{(n)}]'.
$$

Substituting into the (inverse) secant condition we see that

$$
b_n \langle w^{(n)}, q^{(n)} \rangle w^{(n)} = \delta^{(n)} - P_n q^{(n)}.
$$

Thus, the natural choice for $w^{(n)}$ is

$$
w^{(n)} = \delta^{(n)} - P_n q^{(n)},
$$

in which case we must have

$$
b_n = 1/\langle w^{(n)}, q^{(n)} \rangle.
$$

Combining these formulas leads to the \textbf{quasi-Newton} algorithm shown in Table 11.1, also known as the \textbf{Broyden method} [2, p. 77]. One can incorporate a line-search [85] or step-halving procedure into (11.9.1).

Convergence conditions are discussed in [86, 87] and [2, p. 77]. The asymptotic convergence rates of QN can be cumbersome to analyze [2, p. 78]. See [87] for cases where QN algorithms converge Q-superlinearly.

\[\textbf{Mat}\]

The MATLAB function \texttt{fminunc} is based on a quasi-Newton method.

\textbf{Example 11.9.1} Fig. 11.9.1 illustrates the application of the quasi-Newton algorithm to the same problem as in Example 11.3.9, with $P_0 = \frac{\delta}{2} I$. Because the cost function is quadratic, and because $n_p = 2$, after two rank-one updates the inverse Hessian $P_2$ is exactly $H^{-1}$, so $x^{(3)} = \hat{x}$, i.e., convergence in $n_p + 1$ iterations as expected theoretically [18, p. 138].
\[
\begin{align*}
x^{(n+1)} &= x^{(n)} - P_n \nabla \Psi(x^{(n)}) \\
\delta^{(n)} &= x^{(n+1)} - x^{(n)} \\
q^{(n)} &= \nabla \Psi(x^{(n+1)}) - \nabla \Psi(x^{(n)}) \\
w^{(n)} &= \delta^{(n)} - P_n q^{(n)} \\
b_n &= \frac{1}{\langle w^{(n)}, q^{(n)} \rangle} \\
P_{n+1} &= P_n + b_n w^{(n)} [w^{(n)}]' .
\end{align*}
\] (11.9.1)

Table 11.1: Quasi-Newton algorithm.

<table>
<thead>
<tr>
<th>x_2</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Figure 11.9.1: Illustration of a quasi-Newton algorithm.

### 11.9.1 Implementation

A literal implementation using the update (11.9.2) would have prohibitive memory requirements. The only use of \(P_n\) in the algorithm is the operation of multiplying by vectors in \(\mathbb{R}^{n_p}\), which one can implement efficiently as follows:

\[
P_n x = \left[ P_0 + \sum_{i=1}^{n-1} b_i w^{(i)} [w^{(i)}]' \right] x = P_0 x + \sum_{i=1}^{n-1} b_i \langle w^{(i)}, x \rangle w^{(i)}. \] (11.9.3)

As long as \(n \ll n_p\), which is the usual case in imaging problems, the preceding implementation will require much less storage than (11.9.2).

An alternative update strategy is the Broyden, Fletcher, Goldfarb, and Shanno (BFGS) method, for which limited-memory versions are available, e.g., [88]. There are two reasons for limited-memory versions. The obvious reason is to reduce memory requirements, because (11.9.3) requires storage of all \(n\) previous \(w^{(i)}\) vectors. A second reason is that for nonquadratic cost functions, the quadratic approximation (11.4.1) may be poor for the initial \(x^{(n)}\) values, so “forgetting” the effects of those \(x^{(n)}\) values as \(n\) decreases may be helpful.

Another subtle practical consideration is the need to monitor whether \(P_n\) remains positive definite, because the \(b_n\) values can be negative [18, p. 137]. The BFGS method may mitigate this concern [1, p. 63].

Choice of the initial inverse Hessian approximation \(P_0\) is critical [18, p. 137]. The (diagonal) Hessian matrices corresponding to the separable paraboloidal surrogates discussed in subsequent chapters are particularly appealing because those choices ensure that at least the first update (11.9.1) will decrease \(\Psi\). This idea stems from Lange’s proposal to use the EM complete-data information matrix [89].
General-purpose constrained optimization methods

All of the methods described above have been focused on unconstrained optimization. We now turn to the more challenging situation of constrained optimization, where our goal is to compute

$$\hat{x} = \arg \min_{x \in C} \Psi(x),$$

where $C \subset \mathbb{R}^n$ is a given constraint set. The primary types of constraints of interest in image reconstruction are box constraints, defined in Example 27.9.3 and nonnegativity constraints, defined in Example 27.9.4 and examined in Theorem 27.11.1 in §27.11 using the KKT conditions (27.11.1). Occasionally, such as in certain maximum entropy methods, one also sees constraints on the sum of the $x_j$ values:

$$C = \{ x \geq 0 : \sum_{j=1}^n x_j = 1 \}.$$

These constraints are convex sets. In general the minimizer of $\Psi(x)$ may not be unique even over $C$, so we also define

$$X^*(x) = \{ x^* \in C : \Psi(x^*) \leq \Psi(x), \ \forall x \in C \}.$$

When $\Psi$ is a convex cost function and $C$ is a convex set, these constrained minimizers are characterized by Theorem 27.9.25 in §27.9.3.8. However, the following sufficient condition is more suggestive of an iterative algorithm.

**Lemma 11.9.2** [2, p. 209] Let $\Psi$ be a differentiable convex cost function over a convex set $C$. If

$$x^{(*)} = P_C(x^{(*)} - \alpha \nabla \Psi(x^{(*)}))$$

for some $\alpha > 0$, then $x^{(*)} \in X^*(x)$, where $P_C(\cdot)$ is the projector onto the convex set $C$, as defined in (27.9.4) in §27.9.2.

Proof:

From (27.9.3) of Theorem 27.9.5, if $x^{(*)} = P_C(x^{(*)} - \alpha \nabla \Psi(x^{(*)}))$ then

$$\langle (x^{(*)} - \alpha \nabla \Psi(x^{(*)})) - P_C(x^{(*)} - \alpha \nabla \Psi(x^{(*)})), x - P_C(x^{(*)} - \alpha \nabla \Psi(x^{(*)})) \rangle \leq 0, \quad \forall x \in C,$$

or equivalently

$$\langle (x^{(*)} - \alpha \nabla \Psi(x^{(*)})) - x^{(*)}, x - x^{(*)} \rangle \leq 0, \quad \forall x \in C,$$

and hence $\nabla \Psi(x^{(*)}) (x - x^{(*)}) \geq 0, \quad \forall x \in C$, so $x^{(*)} \in X^*(x)$ by Theorem 27.9.3.8. \qed

The relationship (11.9.4) suggests the gradient projection method for constrained minimization, described next.

### 11.10 Gradient projection method (s.opt,convex,fixed)

A simple method for constrained minimization is the gradient projection method; it is essentially the gradient descent method with projection onto the constraint set at the end of each iteration [15, 90–92]:

$$x^{(n+1)} = P_C(x^{(n)} - \alpha \nabla \Psi(x^{(n)})),$$

where $P_C$ denotes the projector onto the convex set $C$ defined in (27.9.4).

At least for simple convex sets such as box constraints, the above algorithm is very simple to implement (see Example 27.9.6). If $\alpha$ is chosen appropriately, then for certain families of cost functions the gradient projection method (11.10.1) converges, as established by Theorem 11.10.1 below. Typically the convergence rate of the gradient projection method is too slow to be very useful in imaging problems. The importance of this theorem and its proof is in the potential applicability of its methods in establishing convergence of other algorithms, such as relaxed forms of ordered subsets algorithms (see §12.10). Bertsekas [1, p. 83] analyzes a generalization of the above algorithm.

The remainder of this section addresses convergence of the gradient projection method (11.10.1), using lemmas established in §27.9.3. We also use the following useful inequality which is easily established by expanding $\|u \pm v/2\|^2 \geq 0$:

$$\|\langle u, v \rangle\| \leq \|u\|^2 + \frac{1}{4} \|v\|^2.$$
Theorem 11.10.1 Let $C$ denote a nonempty, convex, closed subset of $\mathbb{R}^n$. Let $\Psi : \mathbb{R}^n \to \mathbb{R}$ be convex and differentiable with gradient $g(x) \triangleq \nabla \Psi(x)$ satisfying a Lipschitz condition of the form (27.8.5) on $C$. Suppose the set of minimizers $\mathcal{X}(\alpha) = \{x^{(\alpha)} \in C : \Psi(x^{(\alpha)}) \leq \Psi(x), \forall x \in C\}$ is nonempty. If $0 < \alpha < 2/L$, then the gradient projection algorithm (11.10.1) converges to some $x^{(\alpha)} \in \mathcal{X}(\alpha)$.

Proof[2, p. 207]:

Pick any $\hat{x} \in \mathcal{X}(\alpha) \subset C$. By (27.9.3), $(z^{(n)} - P_C(z^{(n)}), \hat{x} - P_C(z^{(n)})) \leq 0$ where $z^{(n)} \triangleq x^{(n)} - \alpha g(x^{(n)})$, hence

$$0 \geq (z^{(n)} - \alpha g(x^{(n)}) - x^{(n+1)}, \hat{x} - x^{(n+1)}).$$

Also, by Theorem 27.9.25

$$0 \geq \alpha \langle g(\hat{x}), \hat{x} - x^{(n+1)} \rangle$$

because $x^{(n+1)} \in C$. Adding and separating yields

$$0 \geq (x^{(n)} - x^{(n+1)}, \hat{x} - x^{(n+1)}) - \alpha \langle g(x^{(n)}), \hat{x} - x^{(n+1)} \rangle.$$

Now

$$\|x^{(n)} - \hat{x}\|^2 = \|x^{(n)} - x^{(n+1)}\|^2 + \|x^{(n+1)} - \hat{x}\|^2 + 2 \langle x^{(n)} - x^{(n+1)}, x^{(n+1)} - \hat{x} \rangle,$$

so substituting into the above inequality and rearranging yields

$$\|x^{(n+1)} - \hat{x}\|^2 \leq \|x^{(n)} - \hat{x}\|^2 - \|x^{(n+1)} - x^{(n)}\|^2 + 2 \alpha \langle g(x^{(n)}), \hat{x} - x^{(n+1)} \rangle$$

$$= \|x^{(n)} - \hat{x}\|^2 - \|x^{(n+1)} - x^{(n)}\|^2 - 2 \alpha \langle g(x^{(n)}), x^{(n)} - \hat{x} \rangle$$

$$+ 2 \alpha \langle g(x^{(n)}), x^{(n)} - x^{(n+1)} \rangle$$

$$\leq \|x^{(n)} - \hat{x}\|^2 - \|x^{(n+1)} - x^{(n)}\|^2 - 2 \alpha \langle g(x^{(n)}), x^{(n)} - \hat{x} \rangle$$

$$+ 2 \alpha \left(1 - \frac{L}{2}\right) \|g(x^{(n)}) - g(\hat{x})\|^2$$

$$\leq \|x^{(n)} - \hat{x}\|^2 - \left(1 - \frac{L}{2}\right) \|x^{(n+1)} - x^{(n)}\|^2,$$

by first applying (11.10.2) with $u = L^{-1/2} (g(x^{(n)}) - g(\hat{x}))$ and $v = L^{1/2} (x^{(n)} - x^{(n+1)})$, and then applying (27.9.13) of Lemma 27.9.21.

Thus $\|x^{(n)} - \hat{x}\|$ is monotone nonincreasing and hence convergent. So $\{x^{(n)}\}$ is bounded, and thus has a subsequence that converges to some limit $x^{(\ast)} \in C$. Furthermore, one can show from the last two inequalities above that $\|x^{(n+1)} - x^{(n)}\| \to 0$.

To show that $x^{(\ast)} \in \mathcal{X}(\ast)$, consider the mapping $T(x) = P_C(x - \alpha \nabla \Psi(x))$. Because $\nabla \Psi(\cdot)$ satisfies a Lipschitz condition it is continuous. The projector $P_C$ is also continuous [93], so $T(\cdot)$ is continuous. So $\|T(x^{(\ast)}) - x^{(\ast)}\| = \lim_{n \to \infty} \|T(x^{(n)}) - x^{(n)}\| = 0$ and hence $T(x^{(\ast)}) = x^{(\ast)}$, so $x^{(\ast)} \in \mathcal{X}(\ast)$ by Lemma 11.9.2.

The limit point $x^{(\ast)}$ is independent of which $\hat{x}$ was considered in the first place, so we may as well have chosen $x^{(\ast)}$, which then shows $\|x^{(n)} - x^{(\ast)}\| \to 0$ because $\|x^{(n)} - x^{(\ast)}\|$ is monotone nonincreasing and the subsequence converges to $x^{(\ast)}$.

Interestingly, this convergence proof does not require monotonicity of $\{\Psi(x^{(n)})\}$. For an alternate proof, see [94].

One can establish $O(1/n)$ rate of convergence of $\Psi(x^{(n)}) - \Psi(\hat{x})$ akin to that in (11.3.16) [15, p. 18]. For an exact worst-case convergence rate in the case where $\Psi$ is strongly convex, see [95].

11.10.1 Example (relation to SQS) (s.opt.gp.sqs)

Consider the WLS cost function $\Psi(x) = \frac{1}{2} \| y - A x \|_2^2$, and the constraint set $C = \{x \in \mathbb{R}^n : x \geq 0\}$, the nonnegative orthant. For this set, $P_C(x) = [x]_+$. The gradient of this cost function is $g(x) = \nabla \Psi(x) = A'W(Ax - y)$ so $\|g(x) - g(\hat{x})\| = \|F(x - \hat{x})\| \leq \|F\| \|x - \hat{x}\|$ where $F \triangleq A'WA$. Clearly the gradient is Lipschitz continuous with $L = \|F\|$. Thus if $0 < \alpha < 2/\|F\|$, then the following GP algorithm is convergent:

$$x^{(n+1)} = [x^{(n)} + \alpha A'W(y - Ax^{(n)})]_+.$$

This is a special case of the separable quadratic surrogate method derived in §14.12.
11.10.2 Limitations to preconditioning (s,opt,gp,pre)

In general one cannot apply just any preconditioner to the gradient projection method [2, p. 210] [96]. The algorithm
\[
\mathbf{x}^{(n+1)} = \mathcal{P}_C(\mathbf{x}^{(n)} - \alpha \mathbf{P} \nabla \Psi(\mathbf{x}^{(n)}))
\]
(11.10.3)

for, say, positive definite \( \mathbf{P} \), will not necessarily converge to some minimizer \( \mathbf{x}^* \in \mathcal{X}^* \), even for small \( \alpha \) [1, p. 77]. A concrete example is given below, where \( \mathbf{P} \) is taken to be the Hessian of \( \Psi \).

When \( \mathcal{C} \) is simply the nonnegative orthant, certain diagonal matrices can be used as preconditioners [1, p. 78] [97]. An example relevant to image reconstruction is the WLS+SQS algorithm given in §14.12 which is of the above form for box constraints and is shown to be convergent in Chapter 13. Modified versions of (11.10.3) called projected Newton methods with more complicated step-size rules can be shown to converge [98]. See also the projected quasi-Newton method [99].

Example 11.10.2 Consider the quadratic cost function \( \Psi(\mathbf{x}) = \frac{1}{2} \| \mathbf{y} - \mathbf{A} \mathbf{x} \|^2 \), with \( \mathbf{A} = \begin{bmatrix} 1 & 0 \\ 1/2 & 1 \end{bmatrix} \) and \( \mathbf{y} = (-3, 0.5) \), and the constraint set \( \mathcal{C} = \{ \mathbf{x} \in \mathbb{R}^2 : \mathbf{x} \geq 0 \} \). The gradient of this cost function is Lipschitz with \( \mathcal{L} = \frac{1 + \sqrt{17}}{8} \approx 1.64 \) the maximum eigenvalue of \( \mathbf{H} = \mathbf{A}' \mathbf{A} \). Consider the preconditioner \( \mathbf{P} = \mathbf{H}^{-1} \), akin to the Newton-Raphson algorithm. Although this preconditioner is ideal in the unconstrained case, it fails completely in this simple example, regardless of \( \alpha \).

Fig. 11.10.1 shows contours of the cost function \( \Psi \), and the trajectory of \( \{ \mathbf{x}^{(n)} \} \) (starting with \( \mathbf{x}^{(0)} = (-2, -2) \)) for \( \alpha = 3/4 \). The algorithm gets “stuck” at \( \mathbf{x}^{(\infty)} = (0, 2) \), which differs from \( \mathbf{x}^* = (0, 1/2) \), the constrained minimizer.

![Figure 11.10.1: Failure of clipped (i.e., projected) Newton-Raphson with a nonnegativity constraint. The larger (yellow) dots show \( \mathbf{x}^{(n)} \) at the start of each iteration. The smaller (green) dots show \( \mathbf{x}^{(n)} - \alpha \mathbf{P} \nabla \Psi(\mathbf{x}^{(n)}) \) prior to zeroing the negatives. The magenta pentagon indicates the location of the constrained minimizer \( \mathbf{x}^* \). The cyan square indicates the unconstrained minimizer at \((-3, 2)\).](fig_wls_nr_clip)

11.11 Newton’s method (s,opt,con,nr)

Consider the problem of minimizing a cost function \( \Psi \) subject to the nonnegativity constraint (11.1.4). A naive implementation of Newton’s method would be the following:

\[
\mathbf{x}^{(n+1)} = \left[ \mathbf{x}^{(n)} - \left( \nabla^2 \Psi(\mathbf{x}^{(n)}) \right)^{-1} \nabla \Psi(\mathbf{x}^{(n)}) \right]_+.
\]

However, usually this method fails to converge to the correct solution, even for a simple quadratic cost function.

Example 11.11.1 For the quadratic cost function \( \Psi(\mathbf{x}) = \frac{1}{2} \mathbf{x}' \mathbf{H} \mathbf{x} - \mathbf{b}' \mathbf{x} \), the above iteration becomes \( \mathbf{x}^{(n+1)} = \left[ \mathbf{x}^{(n)} - \mathbf{H}^{-1} (\mathbf{H} \mathbf{x}^{(n)} - \mathbf{b}) \right]_+ = [\mathbf{H}^{-1} \mathbf{b}]_+ \). So in one iteration the algorithm becomes stuck at a point that is often suboptimal.

See [100] for a quadratic example with equality constraints.

To address this problem, one could use the following more complicated form of Newton’s method for constrained problems:

\[
\mathbf{x}^{(n+1)} = \arg \min_{\mathbf{x} \geq 0} \Psi_n(\mathbf{x}),
\]
(11.11.1)
where $\Psi_n$ is the quadratic approximation defined in (11.4.1). Although one can prove local convergence of this algorithm [2, p. 215], and global convergence for strongly convex cost functions [15, p. 27], it is quite impractical for imaging problems due to the need for constrained minimization of $\Psi_n$ within each iteration. This is a nontrivial optimization problem itself, and is discussed in detail in §14.9.3. See also the projected Newton method [101, p. 159] that modifies the Hessian each iteration based on the active set of constraints.

11.12 Conjugate gradient method \((s,\text{opt,con,cg})\)

One can also develop constrained versions of the conjugate gradient method [2, p. 219] [102] and quasi-Newton methods [2, p. 216]. See §14.11 for the conjugate gradient algorithm with nonnegative constraints for quadratic cost functions.

For problems with nonnegativity constraints, algorithms like the following has been used, e.g., [103].

**CG algorithm with nonnegativity constraints**

**Initialization:**
- Find properly-scaled initial guess $x^{(0)} \succeq 0$.
- Set threshold $\varepsilon \approx 1/1000 \max_j x_j^{(0)}$.
- For each iteration $n = 1, 2, \ldots$:
  - Choose search direction $d^{(n)}$ using the PCG approach (11.8.14).
  - If $x_j^{(n)} < \varepsilon$ and $d_j^{(n)} < 0$, set $d_j^{(n)} = 0$, to avoid leaving the nonnegative orthant. (11.12.1)
  - $d^{(n)} = z^{(n)}_+ - x^{(n)}$.
  - $\alpha_n = \arg\min_{\alpha \in [0,1]} \Psi(x^{(n)} + \alpha d^{(n)})$.
  - $x^{(n+1)} = x^{(n)} + \alpha_n d^{(n)}$.

The step (11.12.1) may not be essential in some applications [103]. The second line search, in (11.12.2), ensures that the iterates $\{x^{(n)}\}$ remain within the nonnegative orthant. However, complications with this approach are the following: truncating the search direction may destroy conjugacy of the search directions, having to do two line searches increases per-iteration computation relative to intrinsically monotone methods.

This is by no means the only possible approach. Active-set methods have also been used in imaging problems, e.g., [104–108]. See also the gradient-projection CG method [101, p. 161].

11.13 Coordinate descent methods \((s,\text{opt,cd})\)

All of the preceding iterations update all elements of $x$ simultaneously. In contrast, another simple and natural approach to finding the minimizer of $\Psi$ is to sequentially minimize $\Psi(x)$ over each element $x_j$ of $x$ using the most recent values for all other elements of $x$. A general coordinate descent (CD) method\(^9\) has the following form:

**Coordinate Descent (CD) “Algorithm”**

\[
\begin{align*}
\text{for } n = 0, 1, \ldots \{ \\
\text{for } j = 1, \ldots, n_p \{ \\
\quad x_j^{(n+1)} := \arg\min_{x_j \geq 0} \Psi(x_1^{(n+1)}, \ldots, x_j^{(n+1)}, x_{j-1}^{(n)}, x_{j+1}^{(n)}, \ldots, x_{n_p}^{(n)}) .
\}
\end{align*}
\]

\(9\) A better name would be coordinate-wise minimization. This method is also called nonlinear Gauss-Siedel and the method of alternating variables [110, p. 53].
The operation in (11.13.1) is performed “in place,” i.e., the new value of \( x_j \) replaces the old value, so that the most recent values of all elements of \( \mathbf{x} \) are always used. An early use of such a method for tomography was in [111].

The coordinate descent approach is often characterized as “very inefficient” in the general optimization literature (e.g., [112, p. 413]), and its computational requirements have been reported incorrectly in the image reconstruction literature (e.g., [113]). Sauer and Bouman analyzed such algorithms using clever frequency domain arguments [114], and showed that sequential algorithms yield iterates whose high spatial-frequency components converge fastest. This is often ideal for tomography, because we can use a low-resolution FBP image as the initial guess, and then iterate to improve resolution and reduce noise, because these are mostly high spatial-frequency effects. (An exception is in limited-angle tomography where the FBP image can have significant low spatial-frequency errors.) Using a uniform or zero initial image for coordinate descent is a very poor choice because low frequencies can converge very slowly. (This discussion assumes the use of a regularized cost function. Coordinate descent may be inappropriate for ill-conditioned problems without regularization.) Furthermore, with an efficient implementation (see §14.7.2), the computational requirements become (nearly) comparable with many other approaches. Thus the traditional concerns about coordinate descent methods are misguided for tomography. However, there is a significant practical disadvantage that arises in very large problems (e.g., 3D PET): coordinate descent methods essentially require that the system matrix be stored by columns, whereas most other algorithms can accommodate more general storage methods, such as factored forms, e.g., [115].

Because the long string of arguments in (11.13.1) is notationally cumbersome, we use the shorthands

\[
\hat{\mathbf{x}}_{(n,j)} = [x_1^{(n+1)}, \ldots, x_{j-1}^{(n)}, x_j^{(n+1)}, x_{j+1}^{(n)}, \ldots, x_n^{(n)}] \quad (11.13.2)
\]

to denote the vector of the most recent parameter values. When unambiguous, we drop the \( (n,j) \), leaving implicit the dependence of \( \hat{\mathbf{x}} \) on iteration \( n \) and pixel index \( j \).

The general method described by (11.13.1) is not exactly an “algorithm,” because the procedure for performing the 1D minimization is yet unspecified. In practice it is usually impractical to find the exact minimizer, even in the 1D problem (11.13.1), so we settle for methods that approximate the minimizer or at least monotonically decrease \( \Psi \).

### 11.13.1 Coordinate-descent Newton (CD-NR)-Raphson

If \( \Psi(\mathbf{x}) \) were a quadratic functional, then the natural approach to performing the minimization in (11.13.1) would be Bouman’s method, which would yield the exact 1D minimizer in (11.13.1) in one iteration. When \( \Psi(\mathbf{x}) \) in (11.13.1) is nonquadratic, applying Newton’s method to (11.13.1) does not guarantee monotonic decreases in \( \Psi \), but one might still try it anyway and hope for the best. In practice, nonmonotonicity often does not seem to be a problem, as suggested by the success of Bouman et al. with this approach [116, 117]. For such a **coordinate-descent Newton-Raphson** (CD-NR) algorithm, we replace (11.13.1) with the following update:

\[
\hat{x}_j^{(n+1)} = \hat{x}_j^{(n)} - \frac{\frac{\partial}{\partial x_j} \Psi(\mathbf{x})}{\frac{\partial^2}{\partial x_j^2} \Psi(\mathbf{x})} \bigg|_{\mathbf{x} = \hat{\mathbf{x}}} \quad (11.13.3)
\]

The optional \( \left[ \cdot \right]_+ \) operation enforces the nonnegativity constraint.

An alternative approach to ensuring monotonicity would be to evaluate the cost function \( \Psi \) after updating each pixel, and impose a **step-halving** search (cf. §11.3) in the (hopefully relatively rare) cases where the cost function decreases. Unfortunately, evaluating \( \Psi \) after every pixel update would add considerable computational overhead.

### 11.13.2 Asymptotic convergence rate of CD

If \( \Psi \) is twice differentiable, then for \( \mathbf{x} \approx \hat{\mathbf{x}} \) the quadratic approximation (11.3.11) is reasonable. Furthermore, if the CD-NR algorithm converges, then near \( \hat{\mathbf{x}} \) its approximate form is

\[
\hat{x}_j^{(n+1)} \approx \left[ x_j^{(n)} - \frac{\frac{\partial}{\partial x_j} \hat{\Psi}(\mathbf{x})}{\frac{\partial^2}{\partial x_j^2} \hat{\Psi}(\mathbf{x})} \bigg|_{\mathbf{x} = \hat{\mathbf{x}}} \right]
\]

Thus, the asymptotic behavior of the CD-NR algorithm is comparable to that of the coordinate descent algorithm for least-squares problems, as described in §14.7.3. Specifically, from §14.7.3:

\[
\| \mathbf{x}^{(n+1)} - \hat{\mathbf{x}} \|_{\mathbf{H}^{1/2}} \leq \rho(M_{GS}) \| \mathbf{x}^{(n)} - \hat{\mathbf{x}} \|_{\mathbf{H}^{1/2}},
\]

where \( \mathbf{H} = \nabla^2 \hat{\Psi}(\hat{\mathbf{x}}) = \mathbf{L} + \mathbf{D} + \mathbf{L}' \) where \( \mathbf{D} \) is diagonal and \( \mathbf{L} \) is lower triangular, and \( M_{GS} = \mathbf{I} - [\mathbf{D} + \mathbf{L}]^{-1} \mathbf{H} \).

Formal analysis of the convergence rate of “pure” CD (11.13.1) follows as a special case of the analysis of SAGE in [118], and agrees with the above approximation.

Convergence of the coordinate descent method for strictly convex, twice-differentiable cost functions is analyzed in detail in [119], including consideration of box constraints. Powell demonstrates that uniqueness of the “arg min” step is important to ensure convergence [120].

Convergence rate analysis, including constrained cases, is given in [121, 122].
11.13.3 Block coordinate descent

Instead of updating one element of $x$ at a time as described in (11.13.1), the block coordinate descent method groups one or more components together and updates them all simultaneously, and then cycles through these blocks. This method is described in more detail in §12.5.8. Convergence results (for limit points) under weak assumptions are given in [3, p. 268]. See also [122–124] and [125]. Generalizations for non-smooth and non-convex functions are in [126, 127]. Examples include image denoising [128] and image reconstruction [129].
11.14 Other constrained optimization methods

A large variety of other constrained optimization methods are available.

11.14.1 Variable transformation methods

Some constrained optimization problems can be converted into unconstrained optimization problems by expressing the unknown (constrained) parameters as nonlinear functions of unconstrained parameters. For example, to enforce the (open) box constraints

\[ 0 < x_j < 1, \]

we can write

\[ x_j = f(\theta_j) \]

where \( f \) is a sigmoid function such as

\[ f(t) = \frac{1}{1+e^{-t}}. \]

With this parameterization, the constrained problem in \( \Psi \)

\[ \hat{x} = \arg \min_{x: 0 < x_j < 1} \Psi(x) \]

becomes the following unconstrained problem in \( \Phi \):

\[ \hat{x}_j = f^{-1}(\hat{\theta}_j), \quad \hat{\theta} = \arg \min_{\theta \in \mathbb{R}^n} \Phi(\theta), \quad \Phi(\theta) \triangleq \Psi(f(\theta)), \]

where \( f(\theta) = (f(x_1), \ldots, f(x_{np})) \). One can apply gradient descent techniques (and other such optimization methods) to the cost function \( \Phi(\theta) \) by using the chain rule for differentiation:

\[ \nabla_{\theta} \Phi(\theta) = (\nabla_{x} \Psi)(f(\theta)) \nabla_{\theta} f(\theta). \]

Here, because \( f(\theta) \) operates element-wise, \( (\nabla_{\theta} f)(\theta) = \text{diag}\{\dot{f}(\theta_j)\} \) so the gradient simplifies to

\[ \frac{\partial}{\partial \theta_j} \Phi(\theta) = \left( \frac{\partial}{\partial x_j} \Psi(x) \right) \bigg|_{x = f(\theta)} \dot{f}(\theta_j). \]

A potential drawback of such approaches is that often the second derivative of the nonlinear function \( f \) will be nearly zero for large and/or small values of its argument, so regularization will be important as always.

11.14.2 Penalty methods

To minimize a cost function \( \Psi(x) \) subject to inequality constraints of the form \( g(x) \leq 0 \) and equality constraints of the form \( h(x) = 0 \), one option is to replace the constraint by a penalty such as:

\[ \hat{x}_\beta = \arg \min_{x} \Psi(x) + \beta \| \max (g(x), 0) \|^2 + \beta \| h(x) \|^2. \]

Typically, as \( \beta \) increases the solution \( \hat{x}_\beta \) comes closer to satisfying the constraint(s) [130]. Unfortunately, as \( \beta \) increases often the conditioning of the minimization problem degrades [1, p. 3]. Nevertheless, this approach has been used for image reconstruction [56].

Recently, augmented Lagrangian methods have become increasingly popular for constrained optimization problems in imaging.
Special-purpose optimization methods

All of the algorithms described above are applicable to minimizing general differentiable cost functions $\Psi$. We now turn to methods that are suitable to some specific cost functions that are important in imaging problems.

11.15 Projection onto convex sets (POCS) (s,opt,pocs)

All of the algorithms described above address minimization problems like (11.1.3). However, there is also an extensive body of literature on tomographic image reconstruction that is based on a rather different philosophy involving constraints. Each measurement is treated as a constraint on the parameterized object $x$, along with any other known constraints such as nonnegativity, and the problem is to find an estimate $\hat{x}$ that satisfies all of the constraints, e.g., [93, 161–170]. Typically these approaches are not statistical formulations because they disregard the log-likelihood of the measurements. Thus they are poorly suited to problems with noisy measurements; indeed modern commercial iterative methods for PET and SPECT and X-ray CT use likelihood-based approaches rather than POCS/ART methods. Nevertheless, we briefly review such methods for completeness.

The usual framework expresses the constraints as a collection of sets $\{C_1, \ldots, C_m\}$ and then the goal is to find an estimate in their intersection:

$$\hat{x} \in C_1 \cap C_2 \cap \cdots \cap C_m.$$  \hspace{1cm} (11.15.1)

If the intersection is empty, then methods exist for finding an estimate $\hat{x}$ that minimizes some type of weighted sum of the distances between $\hat{x}$ and each set $C_k$, e.g., [171]. If the intersection contains multiple solutions, then methods exist for choosing one of the solutions by minimizing some secondary cost function [172].

Even when there exists a unique element $\hat{x}$ in the intersection, computing $\hat{x}$ is a nontrivial problem. If the sets are closed and convex subsets of a Hilbert space, then a natural approach is to successively project $x$ onto each of the sets, i.e.,

$$P_{C_k}(x) = \arg \min_{z \in C_k} \|z - x\|.$$  

In general, one pass of sequentially projecting $x$ onto each of the sets will not yield the solution $\hat{x}$. Gubin et al. [173] proposed the first general method of projections for finding the common point of the intersection of convex sets by cycling through the sets repeatedly. (Kaczmarz [174] had earlier addressed the special case where the sets are hyperplanes, and even earlier von Neumann considered the case of two subspaces [175, Lemma 22].) See [176]. The Douglas-Rachford method is another approach [177–179].

11.15.1 POCS algorithm (sequential)

A projection onto convex sets (POCS) algorithm [wiki] works as follows:

$$x^{(n+1)} = P_{C_m}(P_{C_{m-1}}(\cdots (P_{C_1}(x^{(n)})))).$$  \hspace{1cm} (11.15.2)

Many generalizations have been proposed; see surveys [180, 181].

11.15.2 POCS algorithm (parallel projections)

The goal (11.15.1) is equivalent to the following constrained minimization problem [3, p. 271]:

$$\arg \min_x \min_{z_1 \in C_1, \ldots, z_m \in C_m} \Psi(x, z_1, \ldots, z_m), \quad \Psi(x, z_1, \ldots, z_m) \triangleq \sum_{i=1}^m \|z_i - x\|^2_2.$$  

We can decrease $\Psi$ monotonically by using a block coordinate descent (BCD) algorithm as described in §11.13.3 in which we alternate between updating $x$ and each of the $z_i$ terms:

$$(z_1^{(n+1)}, \ldots, z_m^{(n+1)}) = \arg \min_{z_1 \in C_1, \ldots, z_m \in C_m} \Psi(x^{(n)}, z_1, \ldots, z_m)$$  

$$x^{(n+1)} = \arg \min_x \Psi(x, z_1^{(n+1)}, \ldots, z_m^{(n+1)}) = \frac{1}{m} \sum_{i=1}^m z_i^{(n+1)}.$$
One can show easily that this BCD approach simplifies to the parallel projection algorithm:

\[
\mathbf{x}^{(n+1)} = \frac{1}{m} \sum_{i=1}^{m} P_{C_i}(\mathbf{x}^{(n)}).
\]  

(11.15.3)

It is not obvious from inspection that the iteration (11.15.3) would converge, but by understanding that it is an example of a BCD algorithm one can establish convergence conditions [3, p. 271].

11.15.3 Discussion

One might wonder if convex sets are the only choice for such algorithms. Generalizations have been proposed that address sets in metric spaces [182]. However, in a finite-dimensional Hilbert space, if a set \( C \) is Chebyshev, i.e., if every point \( x \) has a unique projection onto \( C \), then the set is closed and convex [183]. Thus, closed convex sets seem to be the natural choice for practical methods of successive projections.

Unfortunately, although the POCS framework is well-suited to noiseless problems (such as certain types of filter design [184]), it is much less natural for statistical problems, where the negative log-likelihood of the measurements is the natural cost function, rather than some constraint set. Stark et al. [169] used sets based on confidence intervals (derived from the Poisson log-likelihood) for image restoration, but had to choose the confidence parameter empirically to be the natural choice for practical methods of successive projections.

Pocs concepts have been combined with cost functions [185, 186], using proximity operators. The now classical algebraic reconstruction technique (ART) is an example of a specific POCs method, as discussed in the next section.

11.16 ART and its relatives (s,opt,art1)

Most of the iterative algorithms in this book are designed to minimize a certain cost function. However, historically in the field of tomographic image reconstruction, many iterative algorithms have been proposed that were not based on any cost function, but rather were motivated by solving a system of equations \( \mathbf{y} = \mathbf{A}\mathbf{x} \). This section reviews some of those algorithms.

Finding a solution to the system of equations \( \mathbf{y} = \mathbf{A}\mathbf{x} \) is a special case of (11.15.1), i.e., finding a point in the intersection of \( n \) convex sets, where \( C_i \doteq \{ x \in \mathbb{R}^n : [\mathbf{A}x]_i = y_i \} \). (We discard any rows of \( \mathbf{A} \) that are entirely zero.) One can verify using the projection theorem (cf. Theorem 27.9.5) that the projection onto this hyperplane set is

\[
P_{C_i}(\mathbf{x}) = \arg\min_{\mathbf{x} \in C_i} \| \mathbf{z} - \mathbf{x} \| = \mathbf{x} + \frac{y_i - a_i^\top \mathbf{x}}{\| a_i \|^2} a_i = \mathbf{y} - \frac{1}{\| a_i \|^2} a_i (\mathbf{y} - a_i^\top \mathbf{x}),
\]

(11.16.1)

where, \( a_i \) denotes the transpose of the \( i \)th row of the system matrix \( \mathbf{A} \). This property is the foundation of the ART method described next.

11.16.1 ART

The basic (ray sequential) ART algorithm is an iteration of the following form [187, 188] [189, p. 280].

\[
\begin{align*}
\mathbf{x}^{(n,0)} &= \mathbf{x}^{(n)} \\
\mathbf{x}^{(n,i)} &= \mathbf{x}^{(n,i-1)} + a_{n,i}^\top y_i - \frac{[\mathbf{A}\mathbf{x}^{(n,i-1)}]_i}{\| a_i \|^2} a_i \\
\mathbf{x}^{(n+1)} &= \mathbf{x}^{(n,n_d)}.
\end{align*}
\]

(11.16.2)

One loops over each ray index \( i \) in some order. Note that \( \| a_i \|^2 = \sum_{j=1}^{n_d} |a_{ij}|^2 \) and \( [\mathbf{A}\mathbf{x}]_i = (a_i, x) = \sum_{j=1}^{n_d} a_{ij} x_j \). This iteration is also known as Kaczmarz’s method [174, 190]. See [191, p. 277] for a geometric explanation of this method. For consistent data, i.e., \( \mathbf{y} \in \mathcal{R}_A \), and for step sizes \( a_{n,i} \in (0, 2) \), ART converges [192] to the minimum norm solution of \( \mathbf{A}\mathbf{x} = \mathbf{y} \), provided \( \mathbf{x}^{(0)} \in \mathcal{R}_A^\perp \).

11.16.2 MART

The multiplicative ART (MART) algorithm [187], which was designed for entropy optimization, replaces the additive update (11.16.2) with the following multiplicative update [189, p. 284]:

\[
x_{j}^{(n,i)} = \frac{y_i}{(a_i, \mathbf{x}^{(n,i-1)})} a_{ij} x_{j}^{(n,i-1)}.
\]

(11.16.3)
See [189, p. 284] for a convergence theorem with undesirably restrictive conditions. Whereas ART (11.16.2) is applicable even when $A$ has nonnegative (or complex) valued elements, MART clearly needs $a_{ij}$ to be real and nonnegative.

### 11.16.3 SART

Whereas ART updates the image using the data for one value of $i$ at a time, the **simultaneous algebraic reconstruction technique** (SART) uses all data points in a given projection view simultaneously [189, p. 284] [193] [194, p. 82] [191, p. 285]. Let $S_m$, denote the subset of $\{1, \ldots, n_d\}$ corresponding to the $m$th projection view, for $m = 1, \ldots, M$, where $M$ denotes the number of projection views. Letting $A_m$ denote the sub-matrix of $A$ having rows corresponding to $S_m$, and likewise for $y_m$, SART has the following form.

Let $\tilde{c}$ be the vector obtained by normalizing the columns of $A$ to sum to 1. Then $A$ is replaced with $\tilde{A} = \tilde{c}_j A_j$, where $\tilde{c}_j$ is the normalization factor for the $j$th column of $A$. The resulting system can then be solved using an ordinary least squares method.

![Image](https://via.placeholder.com/150)

**Simultaneous Iterative Reconstruction Technique (SIRT)**

In the modern tomography literature, the term “simultaneous” usually means that all pixels are updated simultaneously (as opposed to sequentially like coordinate descent). In that sense of the word, ART is a “somewhat simultaneous” method because many, but not all, pixels are updated simultaneously. For 2D tomography and 3D axial CT, SART is truly a “simultaneous” method because the back-projection of a single view typically influences all (or nearly all) pixels. In helical CT, however, back-projecting a single projection view only influences a portion of the object voxels. So the interpretation of the word “simultaneous” is context dependent. A variation uses multiple views simultaneously, which has been called both ordered subsets SART (OS-SART) [195] OS-SIRT [196] and block SART [197]. See also [198, 199]. Convergence of a modified version of SART, where the leftmost diagonal preconditioning matrix in (11.16.4) is replaced with a subset-independent diagonal matrix, is analyzed in [200]. That diagonal preconditioner could be replaced with an appropriate scalar to save memory if needed [201].

### 11.16.4 SIRT

The **simultaneous iterative reconstruction technique** (SIRT) uses all data points simultaneously to update $x$ at each iteration. The original description in [202] is somewhat confusing, perhaps because it assumes a certain form of the system matrix elements $A$. It was written:

$$x^{(n+1)}_j = \frac{1}{f_j^{(n)}} \left( x^{(n)}_j + \frac{\sum_{i=1}^{n_d} y_i}{\sum_{a_{ij}>0} a_{ij}} - \frac{\sum_{i=1}^{n_d} [A x^{(n)}]_i}{\sum_{a_{ij}>0} a_{ij}} \right),$$

where the normalization factors are included so that the sum of $x^{(n)}_j$ “over all grid points is equal to the sum of” the elements of $x^{(0)}$. A contemporary description (e.g., as given in [201]) is:

$$x^{(n+1)}_j = x^{(n)}_j + \frac{1}{\sum_{i=1}^{n_d} |a_{ij}|} \sum_{i=1}^{n_d} a_{ij} \frac{y_i - [A x^{(n)}]_i}{\sum_{j'=1}^{n_p} a_{ij'}}.$$

(Following [200, 203], absolute values are used here to generalize to cases where $A$ has nonnegative or even complex elements.) With an optional step size $\alpha$, the equivalent matrix-vector form is:

$$x^{(n+1)} = x^{(n)} + \alpha \tilde{C} \tilde{A} \tilde{R} (y - A x^{(n)}),$$  \hspace{1cm} (11.16.5)

where the “**preconditioning matrix**” uses the column sums of $A$:

$$\tilde{C} \triangleq \text{diag} \left\{ \frac{1}{\tilde{c}_j} \right\}, \quad \tilde{c}_j \triangleq \sum_{i=1}^{n_d} |a_{ij}|,$$

and the “weighting” matrix uses the row sums:

$$\tilde{R} \triangleq \text{diag} \left\{ \frac{1}{\tilde{r}_i} \right\}, \quad \tilde{r}_i \triangleq \sum_{j=1}^{n_p} |a_{ij}|.$$

We leave unchanged any pixels where $\tilde{c}_j = 0$. Convergence of SIRT is analyzed in [197, 200, 203, 204]. For necessary and sufficient convergence conditions, see [205]. A generalization where $\tilde{c}_j = \sum_{i=1}^{n_d} |a_{ij}|^p$ and $\tilde{r}_i = \sum_{j=1}^{n_p} |a_{ij}|^{2-p}$

---

*cite sources:*

[189] [193] [194] [191] [197] [200] [203] [204] [205]
with $0 \leq p \leq 2$ is considered in [206]. See [207] for a generalization called the **diagonally-relaxed orthogonal projection (DROP)** method. Pinning down the relationship is an open problem.

The convergence rate of the SIRT iteration (11.16.5) is governed by the eigenvalues of $M = I - \alpha \tilde{C}A'\tilde{RA}$. Because $\tilde{C}A'\tilde{RA}$ is similar to the positive semi-definite matrix $\tilde{C}^{1/2}A'\tilde{R}AC^{1/2}$, the eigenvalues of $\tilde{C}A'\tilde{RA}$ are all real and nonnegative. Thus the eigenvalues of $M$ lie in the real interval $[1 - \alpha \rho(\tilde{C}A'\tilde{RA}), 1]$.

**Lemma 11.16.1** If $\lambda$ is an eigenvalue of $\tilde{C}A'\tilde{RA}$, then $\lambda \leq 1$.

Proof (by contradiction): Suppose $\tilde{C}A'\tilde{RA}x = \lambda x$ for some $x \neq 0$ and some $\lambda > 1$. Without loss of generality we can scale $x$ by $-1$ if necessary so that the element of $x$ with maximum absolute value $|x_j|$ is positive. We can also permute the elements of $A$ so that the first element of $x$ is the largest, i.e., $|x_j| \leq |x_1|$ for all $j$.

By the hypothesis: $|x_1| = \left| \left[ \tilde{C}A'\tilde{RA}x \right]_1 \right| = \frac{1}{\alpha} \left\| \sum_{i=1}^{n_1} a_{i1} \frac{1}{\alpha} \sum_{j=1}^{n_p} a_{ij} x_j \right\| \leq \frac{\sum_{i=1}^{n_1} |a_{i1}| (\frac{1}{\alpha} \sum_{j=1}^{n_p} |a_{ij}| |x_j|)}{\sum_{i=1}^{n_1} |a_{i1}|} \leq \max_i \frac{1}{\alpha} \sum_{j=1}^{n_p} |a_{ij}| |x_j| = \max_j \frac{1}{\alpha} \sum_{i=1}^{n_1} |a_{ij}| |x_i| \leq \max_j |x_j| = x_1$, using the triangle inequality and the fact that a convex combination of numbers is at most their maximum value. This inequality contradicts $\lambda > 1$. □

It follows from this Lemma that $\rho(\tilde{C}A'\tilde{RA}) \leq 1$ and hence if $\alpha < 2$ the eigenvalues of $M$ lie in the interval $(-1, 1)$. Furthermore, if $A$ has full column rank, then the eigenvalues of $M$ lie in the interval $(-1, 1)$.

This proof generalizes slightly the usual argument when $A$ has nonnegative elements, e.g., [201]. When the $a_{ij}$ are all nonnegative, the maximum eigenvalue of $\tilde{C}A'\tilde{RA}$ is exactly 1 and the corresponding eigenvector is $\mathbf{1}_{n_p}$. But in the more general case we know only that $\lambda \leq 1$.

### 11.16.5 Parallel projection version of SIRT

Applying the **parallel projection** algorithm (11.15.3) using (11.16.1) yields the following iteration:

$$x^{(n+1)} = x^{(n)} + \frac{1}{n_1} \sum_{i=1}^{n_1} \frac{1}{\|a_i\|^2} a_i (y_i - a_i^T x^{(n)}) = x^{(n)} + \frac{1}{n_1} A^T \text{diag}\{1/\|a_i\|^2\} (y - Ax^{(n)}).$$

This is yet another variation of the “simultaneous” type of ART method, known as **Cimmino’s algorithm** [207, 208]. One could easily add nonnegativity constraints or box constraints to this algorithm.

### 11.16.6 SIRT for quadratically regularized WLS

If $A$ has full column rank, then the SIRT iteration (11.16.5) converges to the unique minimizer of the geometrically weighted quadratic cost function $f(x) = \frac{1}{2} \|Ax - y\|^2_R$. This is not the desired statistical weighting in problems where noise is problematic\(^{10}\), and lacks regularization. Often we would prefer a cost function of the form $\Psi(x) = \frac{1}{2} \|y - Ax\|_{W^{1/2}}^2 + \frac{1}{2} \|Cx\|_{\tilde{C}^{1/2}}^2$, with statistically based weighting matrix $W$. (We assume any rows where $w_i = 0$ are discarded at the outset.) Inspired by [210–212], define

$$B = \left[ \begin{array}{c} \text{diag}\{p_i\} A \\ \beta \text{diag}\{q_k\} C \end{array} \right], \quad d = \left[ \begin{array}{c} \text{diag}\{p_i\} y \\ 0 \end{array} \right], \quad p_i = w_i \sum_{j=1}^{n_p} |a_{ij}|, \quad q_k = \sum_{j=1}^{n_p} |c_{kj}|.$$

One can verify that applying SIRT (11.16.5) to the (over-determined) system of equations $Bx = d$ yields the iteration

$$x^{(n+1)} = x^{(n)} + \alpha D^{-1} (B^T \, B')^{-1} (B^T \, B) (d - B x^{(n)}) = x^{(n)} - \alpha D^{-1} \Psi'(x^{(n)}),$$

which is equivalent to diagonally preconditioned gradient descent with

$$D = \text{diag} \left\{ \hat{d}_j \right\}, \quad \hat{d}_j = |A'| W |A| 1 + \beta |C'| |C| 1 = \sum_{i=1}^{n_1} w_i |a_{ij}| \left( \sum_{k=1}^{n_p} |a_{ik}| \right) + \beta \sum_{k} \sum_{l=1}^{n_p} |c_{kl}|.$$  

The SIRT iteration (11.16.6) is identical to the SQS algorithm (12.5.13) and (14.6.28) with the $d_j$ defined in (14.6.32). This iteration converges to the unique minimizer of $\Psi$ in the usual case where $WA$ and $C$ have disjoint null spaces [10]. Although the iterations are identical, a potential benefit of the majorization derivation leading to the SQS algorithm is that it generalizes easily to nonquadratic regularizers and readily accommodates nonnegativity constraints.

\(^{10}\) Weighting of the form $w_i = \sum_{j=1}^{n_1} |a_{ij}|^2$ was proposed in [209] without statistical considerations.
11.17 ART2 (s,opt,art2)

In the context of cost functions involving terms of the form $Ax$, such as (1.10.11), coordinate descent algorithms are called column action methods because they use one column of the system matrix $A$ at a time to update an individual pixel. In contrast, the earliest tomographic reconstruction algorithms were row action methods, called, for historical reasons, algebraic reconstruction techniques (ART). Broadly speaking, ART algorithms are special cases of the POCS approach where the constraint sets are hyperplanes. As is typical in POCS methods, measurement noise is not a central component of the problem formulation, i.e., ART methods are designed for problems like “find $x$ such that $y = Ax$” rather than problems like\(^\text{11}\) “find $x$ that minimizes $\|y - Ax\|$.”

A variety of ART algorithms have been proposed, and are discussed at length in classic references e.g., [189, 213–217]. We focus here on ART2 [218], the one particular flavor of ART that comes close to solving the least-squares minimization problem of interest in statistical image reconstruction (see §14.15.5).

The ART2 algorithm [217, p. 152] solves the following linear inequality constraint problem subject to box constraints on $x$:

$$\min_x \frac{1}{2} \|x\|^2 \text{ sub to } \{ \begin{array}{l} \gamma_1 \leq [Ax]_1 \leq \delta_1, \\
\gamma_2 \leq [Ax]_2 \leq \delta_2 \\
x_j^\min \leq x_j \leq x_j^\max, \quad j = 1, \ldots, n \end{array} \}$$

(11.17.1)

where $A$ is a $m \times n$ real matrix.

The nature of these constraints is illustrated in for $m = 2$ in Fig. 11.17.1.

![Figure 11.17.1: Minimizing $\|x\|^2$ subject to linear inequality constraints.](fig_art2)

The ART2 algorithm has the following form.

**ART2 Algorithm**

- Choose $x^{(0)} \in \mathbb{R}^n$ and $z^{(0)} \in \mathbb{R}^m$ such that $x^{(0)} = -A'z^{(0)}$.
- For each $i$ (chosen in some almost cyclic order from $\{1, \ldots, n_d\}$), update $x$ and $z$ as follows:

$$\begin{align*}
x^{(n+1)} &= x^{(n)} + c_n a_i \\
z^{(n+1)} &= z^{(n)} + c_n e_i,
\end{align*}$$

where $a_i$ is the $i$th column of $A'$. The step size is

$$c_n \triangleq \text{med}\left\{ z_i^{(n)}, \frac{\delta_i - \langle a_i, x^{(n)} \rangle}{\|a_i\|^2}, \frac{\gamma_i - \langle a_i, x^{(n)} \rangle}{\|a_i\|^2} \right\},$$

where $\text{med}\{\cdot\}$ denotes the median of the arguments, and where the box-constrained estimate is denoted (with a suggestive “box” symbol):

$$\begin{align*}
x_j^{(n)} &= \begin{cases} x_j^{\min}, & x_j^{(n)} < x_j^{\min} \\
x_j^{(n)}, & x_j^{\min} \leq x_j^{(n)} \leq x_j^{\max} \\
x_j^{\max}, & x_j^{(n)} > x_j^{\max} \end{cases}
\end{align*}$$

It is important to note that $x^{(n)}$ itself does not satisfy the box constraints each iteration, whereas $\tilde{x}^{(n)}$, by definition, always satisfies those constraints. This subtle but critical point has been overlooked in some incorrect descriptions of

\(^{11}\) There is opportunity for confusion on this point, because some ART algorithms have been shown to converge to “least squares” solutions, but usually these correspond to weighted least squares where the weighting is geometric (distances to hyperplanes) rather than statistical (inverse of measurement variances).
the algorithm, e.g., [113]. However, in the limit as the iteration $n$ increases, $x^{(n)}$ will asymptotically satisfy both the box constraints and the other inequality constraints, by the convergence proof of ART2 [217, p. 152].

In the case of equality constraints, where $\delta_i = \gamma_i$, the step size formula simplifies to

$$e_n = \frac{\gamma_i - \langle a_i, x^{(n)} \rangle}{\|a_i\|^2},$$

and $z$ becomes irrelevant, except for the initial condition that $x^{(0)} \in R_\Lambda$. This is essentially the classical ART algorithm [191, p. 283], which does not model measurement noise.

Variations of ART algorithms that are also row-based but include a cost function have also been proposed [219]. Certain duality-based methods also have a row-based flavor [220].

### 11.18 Annealing methods (s,opt,anneal)

Virtually all of the algorithms described above are descent methods. The iterates $\{x^{(n)}\}$ gradually decrease the cost function $\Psi$ each iteration, and, if $\Psi$ is strictly convex and therefore has a unique minimizer $\hat{x}$, most of the algorithms described above will eventually converge to $\hat{x}$.

For cost functions with multiple local minima, descent methods typically converge to a local minimizer that is near the initial guess $x^{(0)}$. In some situations, this local convergence may in fact be adequate if $x^{(0)}$ is a reasonably good guess at the outset and perhaps if there are relatively few local minima of $\Psi$.

However, in situations where $\Psi$ has many local extrema or where a good initial guess is unavailable, alternative optimization strategies are required. One such strategy is simulated annealing, which can “guarantee” (in a probabilistic sense) convergence to a global minimizer of $\Psi$, but with the price of requiring a very large number of iterations. An alternative strategy is deterministic annealing, which does not guarantee global convergence, but often yields good local minimizers in fewer iterations than simulated annealing.

#### 11.18.1 Simulated annealing

The simulated annealing method was originally designed for optimization problems with discrete parameters, such as the famous traveling salesman problem [221]. In such problems there is no gradient so “steepest descent” is inapplicable. The majority of image reconstruction problems involve continuous parameters. A notable exception is problems related to image segmentation, such as with attenuation maps for PET and SPECT attenuation correction. Some formulations of that reconstruction problem assume a discrete set of attenuation coefficients, either from the projection domain, e.g., [222–224], or as an image post-processing procedure e.g., [225, 226]. When such problems are posed as optimization problems, a method like simulated annealing would be needed to ensure global convergence.

Curiously, simulated annealing has been applied even to ordinary least-squares problems, i.e., convex problems in continuous parameters e.g., [227–229], as well as to the ML emission tomography problem [230]. For these problems that have no local minimum, much better methods exist!

A concise summary of how one can adapt the Metropolis algorithm [231] to perform annealing is given in [112], as well as extensions of the method to continuous parameter problems.

An alternative to simulated annealing is the covariance matrix adaptation evolution strategy (CMA-ES) [wiki].

#### 11.18.2 Deterministic annealing and graduated nonconvexity

Chapter 12 illustrates how to simplify the optimization problem of a difficult cost function $\Psi$ by iteratively replacing it with a sequence of more easily minimized surrogate functions. One can apply a similar philosophy to multimodal cost functions $\Psi$, by replacing $\Psi$ with a sequence of “better behaved” cost functions $\{\Psi_k\}, k = 1, 2, \ldots$. This type of approach is called deterministic annealing or graduated nonconvexity and has been applied successfully in a variety of imaging problems, e.g., [232–240].

#### 11.18.3 Other global optimization methods

There is a rich literature on global optimization methods. For differentiable functions, one interesting approach is multilevel coordinate search [241] which combines global search (splitting boxes with large unexplored territory) and local search (splitting boxes with good function values). However, such methods are likely to be impractical for large-scale imaging problems.

#### 11.19 Problems (s,opt,prob)

**Problem 11.1** Consider the differentiable cost function $\Psi(x) = \frac{1}{1+x^2}$. Determine the (smallest possible) Lipschitz constant $L$ for its derivative $\hat{\Psi}$. What happens if we apply GD with $\alpha = 1/L$?
Problem 11.2 Generalize the monotone convergence in norm theory of PGD in §11.3.2 to cases where $P$ is not necessarily Hermitian positive definite.

(Solve?)

Problem 11.3 The PGD method is applied to the quadratic cost function $\Psi(x) = \frac{1}{2} \|y - Ax\|^2 + \beta \frac{1}{2} \|Cx\|^2$ where $F = A^T A$ and $R = C^T C$ are both circulant. The preconditioner is simply $P = I$ and suppose the optimal step size $\alpha$, is used from §11.3.3. Analyze (11.3.12) in the frequency domain to determine which spatial frequency components converge quickly and slowly. As an example, consider the case of Tikhonov regularization where $C = I$ and assume that $F$ has a low-pass nature.

Problem 11.4 Prove the PGD convergence theorem Theorem 11.3.7.

Problem 11.5 Prove Lemma 11.3.11 used to establish $O(1/n)$ convergence of PGD.

Problem 11.6 Prove the $O(1/n)$ convergence of PGD in Theorem 11.3.10 using Lemma 11.3.11.

Problem 11.7 Determine a Lipschitz constant $L_f$ for the derivative of the 1D function $f(\alpha) = \Psi(x + \alpha d)$ when $\Psi$ is the following regularized LS cost function: $\Psi(x) = \frac{1}{2} \|y - Ax\|^2 + \beta \|R(x)\|_2 = \sum_k \psi(|C x |_k)$. Assume that $\psi$ is Lipschitz continuous with constant $L_\psi$. As discussed in §11.6, the Lipschitz constant $L_f$ enables a simple descent method for the line search.

Problem 11.8 Use §11.3.8 to derive the preconditioned BB algorithm of §11.7.1.

Problem 11.9 The formula for $\gamma^H$ given in [61, p. 2] was for the case $P = I$. Use the coordinate transformation ideas of §11.3.8 to derive the preconditioned version (11.8.13).

Problem 11.10 Determine whether the fixed-point condition (11.9.4) is necessary, or merely sufficient. (Need typed.)

Problem 11.11 Generalize Lemma 11.9.2 and the gradient projection algorithm (11.10.1) and its convergence proof Theorem 11.10.1 to include a diagonal preconditioner $D$ with positive diagonal elements. (Solve?)

Problem 11.12 As shown in §11.10, if $C$ is a (nonempty, closed) convex set and $\Psi$ is a convex differential function with Lipschitz gradient, then for suitably small $\alpha$, the gradient projection iteration $x^{(n+1)} = \mathcal{P}_C(x^{(n)} - \alpha \nabla \Psi(x^{(n)}))$ converges to a minimizer of $\Psi(x)$ in $C$.

In some applications, computing the projection $\mathcal{P}_C(z)$ requires an iterative approach. But in some cases we can write $C = C_1 \cap C_2$ where $C_1$ and $C_2$ are both convex sets for which $\mathcal{P}_C(z)$ is easy to evaluate, e.g., for the simplex $C_1 = \{x \geq 0\}$ and $C_2 = \{1^T x = 1\}$.

Is there a simple modification of the gradient projection method that also converges to a minimizer of $\Psi(x)$ over $C$? For example, how about this “GP-POCS” algorithm: $x^{(n+1)} = \mathcal{P}_{C_1}(\mathcal{P}_{C_2}(x^{(n)} - \alpha \nabla \Psi(x^{(n)})))$.

It would be preferable to have an algorithm that generalizes to cases where $C$ is the intersection of more than two sets. This problem was motivated originally by the case where when $C$ is the unit simplex, for which it used to be thought that an iterative approach is needed [254]. Now there is a non-iterative (and fast) approach for the simplex [255, 256].

(Solve?)
## 11.20 Bibliography


