Chapter 11

Optimization by General-Purpose Methods

ch,opt

Contents

11.1 Introduction (s,opt,intro) **11.2**
11.1.1 Iterative optimization methods 11.2
11.1.2 The Hessian 11.3
11.1.3 Why so many algorithms? 11.3
11.1.4 Monotonicity (s,opt,mono) 11.3
11.2 Fixed-point iterations (s,opt,fixed) **11.4**
11.3 Preconditioned gradient descent (PGD) algorithms (s,opt,pgd) **11.4**
11.3.1 Monotonicity conditions for PGD 11.4
11.3.2 Convergence in norm 11.6
11.3.3 Local convergence rate (s,opt,pgd,rate) 11.6
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) 11.8
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 11.9
11.4 Newton-Raphson algorithm (s,opt,nr) **11.9**
11.5 Preconditioned steepest descent (PSD) algorithms (s,opt,psd) **11.10**
11.5.1 Orthogonality and search directions 11.10
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) **11.18**
11.9.1 Implementation 11.19
11.10 Block coordinate descent methods (s,opt,bcm) **11.19**
11.10.1 Block coordinate minimization (BCM) 11.20
11.10.2 Block coordinate descent (BCD) 11.20
11.10.3 Iteration-dependent partitions 11.20
11.10.4 Convergence properties 11.21
11.10.5 Coordinate descent methods (s,opt,cd) 11.21

11.10.5.1 Coordinate-descent Newton (CD-NR)-Raphson . . . 11.22
11.10.5.2 Asymptotic convergence rate of CD . . . 11.22
11.11 Annealing methods (s,opt,anneal) . . . **11.22**
11.11.1 Simulated annealing . . . 11.23
11.11.2 Deterministic annealing and graduated nonconvexity . . . 11.23
11.11.3 Other global optimization methods . . . 11.23
11.12 Problems (s,opt,prob) . . . **11.23**
11.13 Bibliography . . . **11.24**

s,opt,intro

11.1 Introduction (s,opt,intro)

Most of the image reconstruction algorithms described in 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.*, [1–6], and survey papers [7–9]. 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 $\hat{\boldsymbol{x}}$ of a cost function $\Psi(\boldsymbol{x})$, where $\Psi : \mathbb{R}^{n_\mathrm{p}} \to \mathbb{R}$. We make two general assumptions about Ψ throughout this *chapter*:

e,opt,diff

$$\text{Assumption 1. } \Psi \text{ is differentiable} \tag{11.1.1}$$

e,opt,finite

Assumption 2. Ψ has a finite **global minimizer**, *i.e.*,

$$\exists\, \hat{\boldsymbol{x}} \in \mathbb{R}^{n_\mathrm{p}} \;:\; -\infty < \Psi(\hat{\boldsymbol{x}}) \leq \Psi(\boldsymbol{x}), \qquad \forall \boldsymbol{x} \in \mathbb{R}^{n_\mathrm{p}}. \tag{11.1.2}$$

Optimization of non-smooth cost functions is more complicated[1]. Assuming that Ψ has a finite minimizer[2] excludes functions like $\exp(-x)$. Even though this particular example is a **strictly convex** function (see §29.9.3), 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[3]:

e,opt,unc

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x} \in \mathbb{R}^{n_\mathrm{p}}} \Psi(\boldsymbol{x}), \tag{11.1.3}$$

or the nonnegativity-constrained version (considered in Chapter 12):

e,opt,con

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x} \succeq \boldsymbol{0}} \Psi(\boldsymbol{x})\,. \tag{11.1.4}$$

In other words, our goal is to find $\hat{\boldsymbol{x}}$ such that $\Psi(\hat{\boldsymbol{x}}) \leq \Psi(\boldsymbol{x})$, $\forall \boldsymbol{x} \in \mathbb{R}^{n_\mathrm{p}}$ or $\forall \boldsymbol{x} \succeq \boldsymbol{0}$. We write "$\hat{\boldsymbol{x}} = \arg\min$" for simplicity but if there are multiple global minimizers the notation "$\hat{\boldsymbol{x}} \in \arg\min$" would be more precise.

In the absence of any constraints such as nonnegativity, a minimizer $\hat{\boldsymbol{x}}$ of a differentiable cost function $\Psi(\boldsymbol{x})$ is necessarily [10, p. 178] a solution of the following system of n_p equations in n_p unknowns:

e,opt,grad=0

$$\nabla \Psi(\boldsymbol{x}) \Big|_{\boldsymbol{x}=\hat{\boldsymbol{x}}} = \boldsymbol{0}, \tag{11.1.5}$$

where $\boldsymbol{0}$ denotes the $n_\mathrm{p} \times 1$ vector of zeros, and ∇ denotes the column gradient vector:

e,opt,cgrad

$$\nabla \Psi(\boldsymbol{x}) \triangleq \begin{bmatrix} \frac{\partial}{\partial x_1} \Psi(\boldsymbol{x}) \\ \vdots \\ \frac{\partial}{\partial x_{n_\mathrm{p}}} \Psi(\boldsymbol{x}) \end{bmatrix}. \tag{11.1.6}$$

[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 §2.4. One often approximates $|x|$ with $\sqrt{x^2 + \varepsilon}$ for some small positive ε, thereby ensuring differentiability. There are also methods for non-smooth optimization problems, such as those involving the 1-norm, as discussed in §1.12.

[2] Actually, it suffices for Ψ to have a finite minimizer over the set of feasible parameters when constraints such as nonnegativity are involved. An alternative but somewhat less general condition would be to assume that Ψ is a **coercive function**, meaning that $\Psi(\boldsymbol{x}) \to \infty$ as $\|\boldsymbol{x}\| \to \infty$.

[3] The notation $\arg\min_{\boldsymbol{x}}$ indicates the (or an) argument $\boldsymbol{x}$ that minimizes the subsequent expression.

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 16) 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 Ψ to find $\hat{\boldsymbol{x}}$.

11.1.2 The Hessian

Some of the algorithms described below use the **Hessian matrix** of the cost function Ψ, either in implementation or in analysis. For a twice-differential cost function Ψ, the **Hessian matrix**

e,opt,hess

$$\boldsymbol{H}(\boldsymbol{x}) \triangleq \nabla^2 \Psi(\boldsymbol{x}) \tag{11.1.7}$$

has j, kth element given by

$$h_{jk}(\boldsymbol{x}) = \frac{\partial^2}{\partial x_j \partial x_k} \Psi(\boldsymbol{x}), \qquad k, j = 1, \ldots, n_{\mathrm{p}}.$$

For later convenience, we denote the Hessian at the nth iteration as follows:

e,opt,Hn

$$\boldsymbol{H}_n \triangleq \boldsymbol{H}(\boldsymbol{x}^{(n)}). \tag{11.1.8}$$

s,opt,why

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)
- Constraints such as nonnegativity ($\boldsymbol{x} \succeq \boldsymbol{0}$)
- 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.

s,opt,mono

11.1.4 Monotonicity (s,opt,mono)

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 $\{\boldsymbol{x}^{(n)}\}$ that decreases Ψ each iteration, *i.e.*, if

e,opt,mono

$$\Psi\left(\boldsymbol{x}^{(n+1)}\right) \leq \Psi(\boldsymbol{x}^{(n)}), \qquad n = 0, 1, 2, \ldots. \tag{11.1.9}$$

Such algorithms generally have desirable convergence properties, and are analyzed in detail in Chapter 14.

,opt,strictly,monotone

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

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.*, $\nabla \Psi(\boldsymbol{x}^{(n)}) \to 0$.

x,opt,mono,fail

Example 11.1.2 *Consider the **convex** and differentiable cost function* $\Psi(x) = (|x| - 1)^2 \, \mathbb{I}_{\{|x| \geq 1\}}$. *The sequence* $x^{(n)} = (-1)^n (1 + 1/n)$ *has the strictly monotone property* $\Psi\left(x^{(n+1)}\right) < \Psi(x^{(n)})$ *and derivative* $\dot{\Psi}(x^{(n)}) \to 0$, *but* $\{x^{(n)}\}$ *does not converge. For more subtle examples of such pathologies see [11, 12].*

Nevertheless, suitable additional assumptions ensure that a monotone algorithm will converge to a minimizer $\hat{\boldsymbol{x}}$ [13]. Thus, monotone algorithms are particularly desirable.

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

We focus primarily on algorithms that are suitable for large-scale optimization problems, thus excluding numerous classical methods like the **Nelder-Mead simplex algorithm** [14] and related **direct-search** methods [15].

s,opt,fixed

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 $\boldsymbol{M}(\boldsymbol{x})$ of the algorithm designer's choice (that in fact may or may not depend on $\boldsymbol{x}$), and then subtracting both sides from $\hat{\boldsymbol{x}}$ yields the equality

e,opt,fixed

$$\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}} - \boldsymbol{M}(\hat{\boldsymbol{x}}) \nabla \Psi(\hat{\boldsymbol{x}}) . \tag{11.2.1}$$

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

e,opt,alg,recurse

$$\boxed{\boldsymbol{x}^{(n+1)} = \boldsymbol{x}^{(n)} - \boldsymbol{M}(\boldsymbol{x}^{(n)}) \nabla \Psi(\boldsymbol{x}^{(n)}) . \qquad (11.2.2)}$$

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] [10, 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 [16, 17]. Much later it was shown that that Richardson-Lucy algorithm is equivalent to the emission E-ML-EM algorithm discussed in §18.5; 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 $\boldsymbol{M}(\boldsymbol{x})$ can diverge and should be avoided.

s,opt,pgd

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

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

e,opt,pgd,alg

$$\boxed{\boldsymbol{x}^{(n+1)} = \boldsymbol{x}^{(n)} - \alpha \boldsymbol{P} \nabla \Psi(\boldsymbol{x}^{(n)}) . \qquad (11.3.1)}$$

For a general cost function Ψ, one may have difficulty selecting $\boldsymbol{P}$ to ensure global convergence of $\{\boldsymbol{x}^{(n)}\}$ or even monotonicity of $\{\Psi(\boldsymbol{x}^{(n)})\}$. However, for the specific cost functions Ψ of interest in statistical image reconstruction, we will see numerous subsequent examples (*e.g.*, (14.4)) of convergent algorithms of the form (11.3.1).

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

11.3.1 Monotonicity conditions for PGD

By assuming properties of Ψ beyond (11.1.1)-(11.1.2), one can specify sufficient conditions on α and $\boldsymbol{P}$ that ensure monotonicity of PGD.

t,opt,pgd,mono,inv

Theorem 11.3.1 *If the gradient of Ψ is $\boldsymbol{S}$-**Lipschitz continuous** per Definition 29.9.17,* i.e.,

e,opt,pgd,lip,inv

$$\left\| \boldsymbol{S}^{-1} \left(\nabla\Psi(\boldsymbol{x}) - \nabla\Psi(\boldsymbol{z})\right) \right\|_2 \leq \left\| \boldsymbol{S}' \left(\boldsymbol{x} - \boldsymbol{z}\right) \right\|_2 , \qquad \forall \boldsymbol{x}, \boldsymbol{z} \in \mathbb{R}^{n_p}, \tag{11.3.2}$$

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

[5]An algorithm is called **locally convergent** if $\boldsymbol{x}^{(n)} \to \hat{\boldsymbol{x}}$ for some nonempty set of initial guesses $\boldsymbol{x}^{(0)}$ that are "sufficiently close" to $\hat{\boldsymbol{x}}$.

where $\boldsymbol{S}$ is an invertible matrix, $0 < \alpha$, and

e,opt,pgd,cond,inv

$$\alpha \boldsymbol{P}' \boldsymbol{S} \boldsymbol{S}' \boldsymbol{P} \prec \boldsymbol{P} + \boldsymbol{P}', \tag{11.3.3}$$

then the PGD algorithm (11.3.1) monotonically decreases Ψ and in fact is ***strictly monotone****.*
Proof (extended from [2, p. 21]):
By the **first-order Taylor series** with remainder (29.8.3) on $\mathbb{R}^{n_{\mathrm{p}}}$ and (29.2.8) on $\mathbb{C}^{n_{\mathrm{p}}}$:

$$\begin{aligned}\Psi(\boldsymbol{x}^{(n)} + \boldsymbol{z}) &= \Psi(\boldsymbol{x}^{(n)}) + \mathrm{real}\left\{ \int_0^1 \langle \nabla \Psi(\boldsymbol{x}^{(n)} + \tau \boldsymbol{z}),\ \boldsymbol{z} \rangle \, \mathrm{d}\tau \right\} \\ &= \Psi(\boldsymbol{x}^{(n)}) + \mathrm{real}\{\langle \nabla \Psi(\boldsymbol{x}^{(n)}),\ \boldsymbol{z} \rangle\} + \mathrm{real}\left\{ \int_0^1 \langle \nabla \Psi(\boldsymbol{x}^{(n)} + \tau \boldsymbol{z}) - \nabla \Psi(\boldsymbol{x}^{(n)}),\ \boldsymbol{z} \rangle \, \mathrm{d}\tau \right\}.\end{aligned}$$

Identifying $\boldsymbol{z} = -\alpha \boldsymbol{P} \boldsymbol{g}$ for $\alpha \in \mathbb{R}$ where $\boldsymbol{g} \triangleq \nabla \Psi(\boldsymbol{x}^{(n)})$, we have from (11.3.1), (11.3.2), and the **Cauchy-Schwarz inequality** (27.4.2):

$$\begin{aligned}\Psi(\boldsymbol{x}^{(n)}) - \Psi\big(\boldsymbol{x}^{(n+1)}\big) &= \mathrm{real}\{\langle \boldsymbol{g},\ \alpha \boldsymbol{P} \boldsymbol{g} \rangle\} + \mathrm{real}\left\{ \int_0^1 \langle \nabla \Psi(\boldsymbol{x}^{(n)} - \tau \alpha \boldsymbol{P} \boldsymbol{g}) - \boldsymbol{g},\ \alpha \boldsymbol{P} \boldsymbol{g} \rangle \, \mathrm{d}\tau \right\} \\ &= \frac{\alpha}{2} \boldsymbol{g}' (\boldsymbol{P} + \boldsymbol{P}') \boldsymbol{g} + \alpha \int_0^1 \mathrm{real}\{\langle \boldsymbol{S}^{-1} (\nabla \Psi(\boldsymbol{x}^{(n)} - \tau \alpha \boldsymbol{P} \boldsymbol{g}) - \boldsymbol{g}),\ \boldsymbol{S}' \boldsymbol{P} \boldsymbol{g} \rangle\} \, \mathrm{d}\tau \\ &\geq \frac{\alpha}{2} \boldsymbol{g}' (\boldsymbol{P} + \boldsymbol{P}') \boldsymbol{g} - \alpha \int_0^1 \left\| \boldsymbol{S}^{-1} (\nabla \Psi(\boldsymbol{x}^{(n)} - \tau \alpha \boldsymbol{P} \boldsymbol{g}) - \boldsymbol{g}) \right\|_2 \|\boldsymbol{S}' \boldsymbol{P} \boldsymbol{g}\|_2 \, \mathrm{d}\tau \\ &\geq \frac{\alpha}{2} \boldsymbol{g}' (\boldsymbol{P} + \boldsymbol{P}') \boldsymbol{g} - \alpha \int_0^1 \|\boldsymbol{S}' \tau \alpha \boldsymbol{P} \boldsymbol{g}\|_2 \, \mathrm{d}\tau \, \|\boldsymbol{S}' \boldsymbol{P} \boldsymbol{g}\|_2 \\ &= \frac{\alpha}{2} \boldsymbol{g}' (\boldsymbol{P} + \boldsymbol{P}' - \alpha \boldsymbol{P}' \boldsymbol{S} \boldsymbol{S}' \boldsymbol{P}) \boldsymbol{g} \geq 0,\end{aligned}$$

where the last inequality follows from (11.3.3) and §27.2.

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

c,opt,pgd,grad,0

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

(The proof follows from the inequality $\Psi(\boldsymbol{x}^{(n)}) - \Psi\big(\boldsymbol{x}^{(n+1)}\big) \geq \frac{\alpha}{2} \langle \boldsymbol{g}^{(n)},\ (\boldsymbol{P} + \boldsymbol{P}' - \alpha \boldsymbol{P}' \boldsymbol{S} \boldsymbol{S}' \boldsymbol{P}) \boldsymbol{g}^{(n)} \rangle$ in the proof of Theorem 11.3.1.)

Preconditioners are often assumed to be (Hermitian) symmetric, but $\boldsymbol{P}$ in (11.3.3) need not be symmetric. It does follow from (11.3.3) that $\boldsymbol{P}$ is positive definite in the restricted sense that $\boldsymbol{x}' \boldsymbol{P} \boldsymbol{x} > 0$ for all $\boldsymbol{x} \neq \boldsymbol{0}$ in $\mathbb{R}^{n_{\mathrm{p}}}$. However, if $\boldsymbol{P}$ is not symmetric, the other properties in §27.2 need not hold.

c,opt,pgd,mono,inv

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

e,opt,pgd,lip,classic

$$\|\nabla \Psi(\boldsymbol{x}) - \nabla \Psi(\boldsymbol{z})\|_2 \leq \mathcal{L} \|\boldsymbol{x} - \boldsymbol{z}\|_2, \qquad \forall \boldsymbol{x}, \boldsymbol{z} \in \mathbb{R}^{n_{\mathrm{p}}}, \tag{11.3.4}$$

where $\mathcal{L}$ is called the ***Lipschitz constant****, and the monotonicity condition (11.3.3) simplifies to the following "classical" condition on the step size* (cf. *(16.5.19)):*

e,opt,pgd,step,classic

$$0 < \alpha < \frac{2}{\mathcal{L}}. \tag{11.3.5}$$

Establishing the Lipschitz condition (11.3.2) or (11.3.4) is perhaps easiest for twice differentiable cost functions; Theorem 29.9.18 shows that if Ψ is twice differentiable, then (11.3.2) holds iff $\|\boldsymbol{S}^{-1} \nabla^2 \Psi(\boldsymbol{x}) \boldsymbol{S}^{-\mathrm{H}}\|_2 \leq 1$, $\forall \boldsymbol{x} \in \mathbb{R}^{n_{\mathrm{p}}}$. If in addition Ψ is convex, then (11.3.2) holds iff $\nabla^2 \Psi(\boldsymbol{x}) \preceq \boldsymbol{S} \boldsymbol{S}'$ by Corollary 29.9.20. In particular, if Ψ is convex and quadratic with Hessian $\boldsymbol{H}$, then (11.3.2) holds with $\boldsymbol{S} = \boldsymbol{H}^{1/2}$. Furthermore, in that convex and quadratic case, if $\boldsymbol{P}$ is symmetric positive definite, then (11.3.3) simplifies to

e,opt,pgd,alf,quad

$$0 < \alpha < \frac{2}{\lambda_{\max}(\boldsymbol{P} \boldsymbol{H})}. \tag{11.3.6}$$

c,opt,pgd,hess

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

x,opt,pgd,cos

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

x,opt,pgd,huber

Example 11.3.6 *The Huber function $\psi(z)$ in (1.10.9) has derivative $\dot{\psi}(z) = \begin{cases} t, & |z/\delta| \leq 1 \\ \delta\,\mathrm{sgn}(z), & \text{otherwise}. \end{cases}$ It is not twice differentiable, so we cannot apply Corollary 11.3.4, but one can show (by enumerating a few cases) that $\dot{\psi}$ is Lipcshitz with $\mathcal{L} = 1$. This example illustrates that the Lipschitz gradient condition Theorem 11.3.1 is more general than curvature bounds like Theorem 29.9.18.*

s,opt,pgd,norm

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 $\{\boldsymbol{x}^{(n)}\}$ approach a minimizer $\hat{\boldsymbol{x}}$ monotonically. To provide sufficient conditions for such **monotone convergence in norm** we focus on cases where Ψ is twice differentiable with positive definite Hessian ($\boldsymbol{0} \prec \nabla^2\Psi$, and hence is strictly convex) with unique minimizer $\hat{\boldsymbol{x}}$ and where $\boldsymbol{P}$ is also positive definite (and thus has an invertible square root, per Definition 27.2.1). Because $\nabla\Psi(\hat{\boldsymbol{x}}) = \boldsymbol{0}$ we use (29.8.4) to rewrite the PGD iteration (11.3.1) as

$$\begin{aligned} \boldsymbol{x}^{(n+1)} - \hat{\boldsymbol{x}} &= \boldsymbol{x}^{(n)} - \hat{\boldsymbol{x}} - \alpha\boldsymbol{P}\left(\nabla\Psi(\boldsymbol{x}^{(n)}) - \nabla\Psi(\hat{\boldsymbol{x}})\right) & (11.3.7)\\ &= \boldsymbol{x}^{(n)} - \hat{\boldsymbol{x}} - \alpha\boldsymbol{P}\left(\int_0^1 \nabla^2\Psi(\hat{\boldsymbol{x}} + \tau\left(\boldsymbol{x}^{(n)} - \hat{\boldsymbol{x}}\right))\,\mathrm{d}\tau\right)\left(\boldsymbol{x}^{(n)} - \hat{\boldsymbol{x}}\right) & (11.3.8)\\ &= \boldsymbol{P}^{1/2}\left(\boldsymbol{I} - \alpha\boldsymbol{P}^{1/2}\left(\int_0^1 \nabla^2\Psi(\hat{\boldsymbol{x}} + \tau\left(\boldsymbol{x}^{(n)} - \hat{\boldsymbol{x}}\right))\,\mathrm{d}\tau\right)\boldsymbol{P}^{1/2}\right)\boldsymbol{P}^{-1/2}\left(\boldsymbol{x}^{(n)} - \hat{\boldsymbol{x}}\right). & (11.3.9) \end{aligned}$$

e,opt,pgd,err

Thus

$$\left\|\boldsymbol{P}^{-1/2}\left(\boldsymbol{x}^{(n+1)} - \hat{\boldsymbol{x}}\right)\right\| \leq \left\|\!\left|\boldsymbol{I} - \alpha\boldsymbol{P}^{1/2}\left(\int_0^1 \nabla^2\Psi(\hat{\boldsymbol{x}} + \tau\left(\boldsymbol{x}^{(n)} - \hat{\boldsymbol{x}}\right))\,\mathrm{d}\tau\right)\boldsymbol{P}^{1/2}\right|\!\right\| \left\|\boldsymbol{P}^{-1/2}\left(\boldsymbol{x}^{(n)} - \hat{\boldsymbol{x}}\right)\right\|, \quad (11.3.10)$$

e,opt,pgd,norm,mono

and using Lemma 27.2.3, if $\alpha\nabla^2\Psi(\boldsymbol{x}) \prec \boldsymbol{P}^{-1}$, $\forall\boldsymbol{x}$, then $\left\|\boldsymbol{P}^{-1/2}\left(\boldsymbol{x}^{(n)} - \hat{\boldsymbol{x}}\right)\right\|$, the *weighted* distance to minimizer $\hat{\boldsymbol{x}}$, decreases every iteration. If $\boldsymbol{P} = \boldsymbol{I}$ 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.

t,opt,pgd,norm

Theorem 11.3.7 *If Ψ is convex and its gradient $\nabla\Psi$ is $\boldsymbol{S}$-Lipschitz per (11.3.2), if the set of minimizers $\mathcal{X}^{(\star)} = \{\boldsymbol{x}^{(\star)} \in \mathbb{C}^{n_p} : \Psi(\boldsymbol{x}^{(\star)}) \leq \Psi(\boldsymbol{x}),\ \forall\boldsymbol{x} \in \mathbb{C}^{n_p}\}$ is nonempty, and if $\boldsymbol{P} = [\boldsymbol{T}\boldsymbol{T}']^{-1}$ where $\boldsymbol{0} \prec \alpha\boldsymbol{S}\boldsymbol{S}' \prec 2\boldsymbol{T}\boldsymbol{T}'$, then $\|\boldsymbol{T}'\left(\boldsymbol{x}^{(n)} - \hat{\boldsymbol{x}}\right)\|$ is monotone nonincreasing for the PGD iteration and $\{\boldsymbol{x}^{(n)}\}$ converges to some $\boldsymbol{x}^{(\star)} \in \mathcal{X}^{(\star)}$.*
Proof:
Follow the proof of Theorem 12.2.1. See Problem 11.4.

x,opt,pgd,mono

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

s,opt,pgd,rate

11.3.3 Local convergence rate (s,opt,pgd,rate)

An algorithm designer must choose the preconditioner $\boldsymbol{P}$, and this choice is aided by analysis of the **asymptotic convergence rate** of the PGD method. If Ψ is twice differentiable in the neighborhood of a local minimizer $\hat{\boldsymbol{x}}$, with Hessian $\hat{\boldsymbol{H}} \triangleq \nabla^2\hat{\Psi}(\hat{\boldsymbol{x}})$, then **Ostrowski's theorem** [18, p. 300], see (29.13.5), states that the **root convergence factor** for sequences converging to $\hat{\boldsymbol{x}}$ is

$$R_1 = \rho(\boldsymbol{I} - \alpha\boldsymbol{P}\hat{\boldsymbol{H}}).$$

To elaborate this rate, we assume that Ψ is locally twice differentiable *i.e.*, for $\boldsymbol{x} \approx \hat{\boldsymbol{x}}$:

e,opt,hkost

$$\Psi(\boldsymbol{x}) \approx \hat{\Psi}(\boldsymbol{x}) \triangleq \Psi(\hat{\boldsymbol{x}}) + \frac{1}{2} (\boldsymbol{x} - \hat{\boldsymbol{x}})' \hat{\boldsymbol{H}} (\boldsymbol{x} - \hat{\boldsymbol{x}}), \tag{11.3.11}$$

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

$$\boldsymbol{x}^{(n+1)} \approx \boldsymbol{x}^{(n)} - \alpha \boldsymbol{P} \nabla \hat{\Psi}(\boldsymbol{x}^{(n)}) = \boldsymbol{x}^{(n)} - \alpha \boldsymbol{P} \hat{\boldsymbol{H}} (\boldsymbol{x}^{(n)} - \hat{\boldsymbol{x}}).$$

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

e,opt,pgd,rate,recurse

$$\boldsymbol{P}^{-1/2} (\boldsymbol{x}^{(n+1)} - \hat{\boldsymbol{x}}) \approx \left(\boldsymbol{I} - \alpha \boldsymbol{P}^{1/2} \hat{\boldsymbol{H}} \boldsymbol{P}^{1/2} \right) \boldsymbol{P}^{-1/2} (\boldsymbol{x}^{(n)} - \hat{\boldsymbol{x}}). \tag{11.3.12}$$

Hence, to within that approximation:

e,opt,pgd,norm

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

(Example 11.3.9 below shows that inequality (11.3.13) is tight.) Thus the **asymptotic convergence rate** of PGD (*cf.* §29.13) is governed by the **spectral radius** $\rho(\boldsymbol{I} - \alpha \boldsymbol{P}^{1/2} \hat{\boldsymbol{H}} \boldsymbol{P}^{1/2})$. Qualitatively speaking, the closer $\alpha \boldsymbol{P}$ is to $\hat{\boldsymbol{H}}^{-1}$, the faster the convergence asymptotically. For a given $\boldsymbol{P}$, the best (asymptotic) step size α is

e,opt,pgd,rate,best

$$\alpha_\star = \frac{2}{\lambda_{\min}(\boldsymbol{P}\hat{\boldsymbol{H}}) + \lambda_{\max}(\boldsymbol{P}\hat{\boldsymbol{H}})}, \tag{11.3.14}$$

provided $\boldsymbol{P}\hat{\boldsymbol{H}}$ has nonnegative eigenvalues (*e.g.*, if $\boldsymbol{P}$ is positive definite), in which case

e,opt,pgd,cond

$$\rho(\boldsymbol{I} - \alpha_\star \boldsymbol{P}\hat{\boldsymbol{H}}) = \frac{\lambda_{\max}(\boldsymbol{P}\hat{\boldsymbol{H}}) - \lambda_{\min}(\boldsymbol{P}\hat{\boldsymbol{H}})}{\lambda_{\max}(\boldsymbol{P}\hat{\boldsymbol{H}}) + \lambda_{\min}(\boldsymbol{P}\hat{\boldsymbol{H}})} = \frac{\kappa - 1}{\kappa + 1}, \tag{11.3.15}$$

where $\kappa \triangleq \lambda_{\max}(\boldsymbol{P}\hat{\boldsymbol{H}}) / \lambda_{\min}(\boldsymbol{P}\hat{\boldsymbol{H}})$ denotes the **condition number** of $\boldsymbol{P}\hat{\boldsymbol{H}}$. For the fastest convergence, one would like to choose $\boldsymbol{P}$ to minimize the condition number of the product $\boldsymbol{P}\hat{\boldsymbol{H}}$.

The approximate analysis above can be made rigorous for **strongly convex** cost functions [19, p. 14] [2, p. 24].

x,opt,pgd,tight

Example 11.3.9 *The inequality (11.3.13) is tight. Consider the (separable!) cost function* $\Psi(\boldsymbol{x}) = \frac{1}{2} \boldsymbol{x}' \boldsymbol{H} \boldsymbol{x}$ *where* $\boldsymbol{H} = \begin{bmatrix} 1 & 0 \\ 0 & \kappa \end{bmatrix}$ *and* $\boldsymbol{P} = \boldsymbol{I}$. *If* $\boldsymbol{x}^{(n)} = c \begin{bmatrix} \pm\kappa \\ 1 \end{bmatrix}$, *then simple algebra shows that* $\boldsymbol{d}^{(n)} = -\kappa c \begin{bmatrix} \pm 1 \\ 1 \end{bmatrix}$ *and* $\alpha_\star = \frac{2}{1+\kappa}$ *and* $\boldsymbol{x}^{(n+1)} = c \frac{\kappa - 1}{\kappa + 1} \begin{bmatrix} \pm\kappa \\ -1 \end{bmatrix}$, *so* $\|\boldsymbol{x}^{(n)}\| = \left(\frac{\kappa-1}{\kappa+1} \right)^n \|\boldsymbol{x}^{(0)}\|$, *which is convergence exactly at the upper bound rate given in (11.3.13). Fig. 11.3.1 displays* $\{\boldsymbol{x}^{(n)}\}$ *and contours of* Ψ *for the case* $\kappa = 4$.

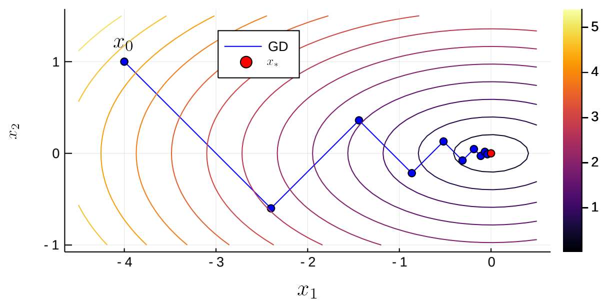


Figure 11.3.1: Illustration of slow convergence of (unpreconditioned) gradient descent for a separable cost function. The iteration begins at $\boldsymbol{x}^{(0)}$ and the iterates $\{\boldsymbol{x}^{(n)}\}$ (indicated by the circles) converge slowly to the minimizer $\hat{\boldsymbol{x}} = \boldsymbol{0}$.

fig_opt_pgd_tight

s,opt,pgd,1n
11.3.4 Convergence rate of cost function decrease: $O(1/n)$ (s,opt,pgd,1n)

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

t,opt,pgd,1n
Theorem 11.3.10 *Suppose Ψ is convex and differentiable, with gradient satisfying a Lipschitz condition of the form (11.3.2), and has a minimizer $\hat{\boldsymbol{x}}$. If we choose $\alpha \boldsymbol{P} = [\boldsymbol{S}\boldsymbol{S}']^{-1}$, then the PGD algorithm (11.3.1) produces a sequence $\{\boldsymbol{x}^{(n)}\}$ for which [20, Thm 3.1] [19]*

$$\Psi(\boldsymbol{x}^{(n)}) - \Psi(\hat{\boldsymbol{x}}) \leq \frac{\left\|\boldsymbol{S}'\left(\boldsymbol{x}^{(0)} - \hat{\boldsymbol{x}}\right)\right\|^2}{2n}, \; n \geq 1. \tag{11.3.16}$$ e,opt,pgd,1n

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 $\{\Psi(\boldsymbol{x}^{(n)})\}$, not $\{\|\boldsymbol{x}^{(n)} - \hat{\boldsymbol{x}}\|\}$.

The proof (see Problem 11.6) relies on the following Lemma that generalizes [20, Lemma 2.3].

l,opt,pgd,1n
Lemma 11.3.11 *(See Problem 11.5.) If Ψ is convex and differentiable, with gradient satisfying a Lipschitz condition of the form (11.3.2), and if we define the PGD update by*

$$M(\boldsymbol{x}) = \boldsymbol{x} - \alpha \boldsymbol{P} \, \nabla\Psi(\boldsymbol{x}), \tag{11.3.17}$$ e,opt,pgd,1n,M

where $\boldsymbol{P}$ satisfies (11.3.3), then

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

where by (11.3.3)

$$\boldsymbol{B} \triangleq \frac{1}{\alpha}\left(\boldsymbol{P}^{-1} + \boldsymbol{P}^{-\mathrm{T}}\right) - \boldsymbol{S}\boldsymbol{S}' \succeq \boldsymbol{0}. \tag{11.3.18}$$ e,opt,pgd,1n,B

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 $\boldsymbol{H}$, by (11.3.11)

$$\begin{aligned}
&\Psi(\boldsymbol{x}^{(n)}) - \Psi(\hat{\boldsymbol{x}}) \\
&\quad = \frac{1}{2}\left(\boldsymbol{x}^{(n)} - \hat{\boldsymbol{x}}\right)' \boldsymbol{H}\left(\boldsymbol{x}^{(n)} - \hat{\boldsymbol{x}}\right) = \frac{1}{2}\left(\boldsymbol{P}^{-1/2}\left(\boldsymbol{x}^{(n)} - \hat{\boldsymbol{x}}\right)\right)' \boldsymbol{P}^{1/2}\boldsymbol{H}\boldsymbol{P}^{1/2}\left(\boldsymbol{P}^{-1/2}\left(\boldsymbol{x}^{(n)} - \hat{\boldsymbol{x}}\right)\right) \\
&\quad \leq \frac{1}{2}\|\boldsymbol{P}^{1/2}\boldsymbol{H}\boldsymbol{P}^{1/2}\| \left\|\boldsymbol{P}^{-1/2}\left(\boldsymbol{x}^{(n)} - \hat{\boldsymbol{x}}\right)\right\|^2 \\
&\quad \leq \frac{1}{2}\|\boldsymbol{P}^{1/2}\boldsymbol{H}\boldsymbol{P}^{1/2}\| \|\boldsymbol{I} - \alpha\boldsymbol{P}^{1/2}\boldsymbol{H}\boldsymbol{P}^{1/2}\|^{2n} \left\|\boldsymbol{P}^{-1/2}\left(\boldsymbol{x}^{(0)} - \hat{\boldsymbol{x}}\right)\right\|^2,
\end{aligned}$$

using (11.3.10). So if $\boldsymbol{0} \prec \alpha\boldsymbol{H} \prec \boldsymbol{P}^{-1}$ then Ψ decreases geometrically, *i.e.*, $O(\rho^n)$, where $\rho = \|\boldsymbol{I} - \alpha\boldsymbol{P}^{1/2}\boldsymbol{H}\boldsymbol{P}^{1/2}\|^2$, rather than at the "sublinear" rate of only $O(1/n)$. More generally, for any **strongly convex** cost function Ψ one can establish a geometric convergence rate for $\|\boldsymbol{x}^{(n)} - \hat{\boldsymbol{x}}\|$ [19, p. 18].

Drori and Teboulle [21] derived a bound like (11.3.16) except with $4n+2$ in the denominator. They also construct a Huber-like function Ψ for which GD achieves that bound. Thus, the $O(1/n)$ rate is tight over the family of cost functions with Lipschitz gradients.

s,opt,pgd,ox
11.3.5 Relationship with optimization transfer (s,opt,pgd,ox)

If $\nabla\Psi$ satisfies the Lipschitz condition (11.3.2), then Ψ has the following quadratic **majorizer** (*cf.* Chapter 14):

$$\Psi(\boldsymbol{x}) \leq \phi^{(n)}(\boldsymbol{x}) \triangleq \Psi(\boldsymbol{x}^{(n)}) + \operatorname{real}\{\langle\nabla\Psi(\boldsymbol{x}^{(n)}), \, \boldsymbol{x} - \boldsymbol{x}^{(n)}\rangle\} + \frac{1}{2}\left\|\boldsymbol{S}'\left(\boldsymbol{x} - \boldsymbol{x}^{(n)}\right)\right\|_2^2. \tag{11.3.19}$$ e,opt,pgd,ox,qs

If we choose $\boldsymbol{P} = [\boldsymbol{S}\boldsymbol{S}']^{-1}$ then (11.3.3) simplifies to $0 < \alpha < 2$. In particular, for $\alpha = 1$ the update becomes

$$\boldsymbol{x}^{(n+1)} = \boldsymbol{x}^{(n)} - [\boldsymbol{S}\boldsymbol{S}']^{-1} \nabla\Psi(\boldsymbol{x}^{(n)}) = \arg\min_{\boldsymbol{x}} \phi^{(n)}(x)\,.$$

Optimization transfer methods in Chapter 14 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 $\alpha\boldsymbol{P} = [\boldsymbol{S}\boldsymbol{S}']^{-1}$.

s,opt,pgd,half

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

The PGD algorithm in general does not ensure that $\Psi(\boldsymbol{x})$ decreases monotonically, and often it is challenging to find an appropriate step size α. However, one can modify the algorithm by incorporating **backtracking** [22, p. 131] to ensure descent. Define the search direction

e,opt,predir

$$\boldsymbol{d}^{(n)} = -\boldsymbol{P}\,\nabla\Psi(\boldsymbol{x}^{(n)}), \tag{11.3.20}$$

and replace $\alpha\boldsymbol{d}^{(n)}$ in (11.3.1) with $\alpha_n\boldsymbol{d}^{(n)}$ where we choose α_n to ensure that

e,opt,pgd,mono

$$\Psi(\boldsymbol{x}^{(n)} + \alpha_n\boldsymbol{d}^{(n)}) \le \Psi(\boldsymbol{x}^{(n)})\,. \tag{11.3.21}$$

In particular, one can apply a simple **step-halving** procedure, where one starts with some initial α_n value and then decreases α_n by a factor of 2 until (11.3.21) is satisfied. Following [22, p. 131], if $\boldsymbol{P}$ is positive definite, then for $\boldsymbol{d} = -\boldsymbol{P}\,\nabla\Psi(\boldsymbol{x}) \neq \boldsymbol{0}$, **Taylor's theorem** yields

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

which will be positive for sufficiently small α, because $\boldsymbol{d}'\boldsymbol{P}^{-1}\boldsymbol{d} > 0$ for positive definite $\boldsymbol{P}$, and $o(\alpha)/\alpha$ approaches zero as $\alpha \to 0$. Thus, (11.3.20) is a **descent direction** at $\boldsymbol{x}^{(n)}$, so step-halving is always guaranteed to lead (eventually) to a value $\boldsymbol{x}^{(n+1)}$ that decreases Ψ. However, obtaining this guarantee incurs the computational price of multiple evaluations of $\Psi(\boldsymbol{x}^{(n)} + \alpha\boldsymbol{d}^{(n)})$.

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 Ψ *without* any **line search** or backtracking.

s,opt,precon

11.3.7 Ideal preconditioner (s,opt,precon)

From the analysis in §11.3.3, for *quadratic* cost functions, the ideal preconditioner would be $\boldsymbol{P}_0 = \boldsymbol{H}^{-1}$ so that $\boldsymbol{P}_0\boldsymbol{H} = \boldsymbol{I}$, because the $n_\mathrm{p} \times n_\mathrm{p}$ identity matrix $\boldsymbol{I}$ has the minimal condition number (unity), and the PSD algorithm would converge in one step. For nonquadratic Ψ, the inverse-Hessian preconditioner $\boldsymbol{P}_0(\boldsymbol{x}) = \boldsymbol{H}^{-1}(\boldsymbol{x})$ would yield superlinear convergence rates akin to the Newton-Raphson method [23]. Because we cannot compute $\boldsymbol{H}^{-1}$ for large n_p, one must develop preconditioners that *approximate* $\boldsymbol{H}^{-1}$; see Chapter 20.

s,opt,precon,change

11.3.8 Preconditioning as a coordinate transformation

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

e,opt,precon,change,kost,P

$$\hat{\boldsymbol{z}} = \arg\min_{\boldsymbol{z}} \Psi_P(\boldsymbol{z}), \qquad \Psi_P(\boldsymbol{z}) \triangleq \Psi(\boldsymbol{T}\boldsymbol{z})\,. \tag{11.3.22}$$

A gradient descent algorithm for Ψ_P in terms of $\boldsymbol{z}$ has the form

$$\boldsymbol{z}^{(n+1)} = \boldsymbol{z}^{(n)} - \alpha\nabla_{\boldsymbol{z}}\Psi_P(\boldsymbol{z}^{(n)}) = \boldsymbol{z}^{(n)} - \alpha\boldsymbol{T}'\,\nabla\Psi(\boldsymbol{T}\boldsymbol{z}^{(n)}),$$

by applying the chain rule. Multiplying by $\boldsymbol{T}$ yields the PGD update (11.3.1). So preconditioning is equivalent to a coordinate change. 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.

s,opt,nr

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

From the preceding convergence analysis, the optimal preconditioner would be $\alpha\boldsymbol{P} = \boldsymbol{H}(\hat{\boldsymbol{x}})^{-1}$ (asymptotically). Because $\hat{\boldsymbol{x}}$ is unknown, this preconditioner is impractical. However, in principle we can achieve comparable performance by using the current Hessian $\boldsymbol{H}_n = \boldsymbol{H}(\boldsymbol{x}^{(n)})$ instead of $\boldsymbol{H}(\hat{\boldsymbol{x}})$. 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 Ψ is twice differentiable, then we can make a 2nd-order **Taylor series** approximation[6] using (11.1.8):

e,opt,taylorn

$$\Psi(\boldsymbol{x}) \approx \Psi_n(\boldsymbol{x}) \triangleq \Psi(\boldsymbol{x}^{(n)}) + \text{real}\{\langle \nabla\Psi(\boldsymbol{x}^{(n)}),\, \boldsymbol{x} - \boldsymbol{x}^{(n)} \rangle\} + \frac{1}{2}\left(\boldsymbol{x} - \boldsymbol{x}^{(n)}\right)' \boldsymbol{H}_n \left(\boldsymbol{x} - \boldsymbol{x}^{(n)}\right). \tag{11.4.1}$$

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

e,opt,taylorn,0

$$\mathbf{0} = \nabla\Psi_n(\boldsymbol{x})\Big|_{\boldsymbol{x}=\boldsymbol{x}^{(n+1)}} = \nabla\Psi(\boldsymbol{x}^{(n)}) + \boldsymbol{H}_n \left(\boldsymbol{x} - \boldsymbol{x}^{(n)}\right)\Big|_{\boldsymbol{x}=\boldsymbol{x}^{(n+1)}}. \tag{11.4.2}$$

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

e,opt,alg,nr

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

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 Ψ. (One can attempt to overcome this limitation using backtracking, see (11.3.21), at least for convex Ψ, but in general adding a line search does not ensure convergence [24].) When Ψ 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: $\|\boldsymbol{x}^{(n)} - \hat{\boldsymbol{x}}\| \leq c q^{2^n}$, for a constant c that depends on Ψ and a constant $q < 1$ that depends on Ψ and $\boldsymbol{x}^{(0)}$ [19, 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 §14.13. For optimal second-order methods for nonconvex problems, see [25].

s,opt,psd

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 $\hat{\boldsymbol{H}}$ depends on the unknown minimizer $\hat{\boldsymbol{x}}$. 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 $\boldsymbol{d}^{(n)}$ defined as the (negative) preconditioned gradient in (11.3.20), the **preconditioned steepest descent** (**PSD**) algorithm [26] is given as follows:

e,opt,psd,line

e,opt,psd,alg

$$\boxed{\begin{aligned} \boldsymbol{d}^{(n)} &\triangleq -\boldsymbol{P}\,\nabla\Psi(\boldsymbol{x}^{(n)}) \\ \alpha_n &\triangleq \underset{\alpha\in[0,\infty)}{\arg\min}\, \Psi(\boldsymbol{x}^{(n)} + \alpha\boldsymbol{d}^{(n)}) && (11.5.1) \\ \boldsymbol{x}^{(n+1)} &= \boldsymbol{x}^{(n)} + \alpha_n \boldsymbol{d}^{(n)}, && (11.5.2) \end{aligned}}$$

where α_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 Ψ, 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

e,opt,psd,gnn,dirn,0

$$0 = \frac{\partial}{\partial\alpha}\, \Psi(\boldsymbol{x}^{(n)} + \alpha\boldsymbol{d}^{(n)})\Big|_{\alpha=\alpha_n} = \langle \nabla\Psi(\boldsymbol{x}^{(n)} + \alpha\boldsymbol{d}^{(n)}),\, \boldsymbol{d}^{(n)} \rangle\Big|_{\alpha=\alpha_n} = \langle \nabla\Psi\big(\boldsymbol{x}^{(n+1)}\big),\, \boldsymbol{d}^{(n)} \rangle, \tag{11.5.3}$$

[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 (29.4.2).

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

11.5.2 Complex case

We can also apply PSD to cost functions with complex-valued arguments, *i.e.*, $\Psi : \mathbb{C}^{n_{\mathrm{p}}} \to \mathbb{R}$, where $\boldsymbol{x}^{(n)}, \boldsymbol{d}^{(n)} \in \mathbb{C}^{n_{\mathrm{p}}}$, provided we define $\nabla\Psi$ appropriately; see (29.2.6) in Appendix 29. The line search (11.5.1) is still over $\alpha \in [0, \infty)$, even when $\boldsymbol{x}$ is complex. For complex cases,

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

and the orthogonality condition (11.5.3) becomes:

e,opt,psd,gnn,dirn,0,complex

$$\mathrm{real}\big\{\langle\nabla\Psi\big(\boldsymbol{x}^{(n+1)}\big),\, \boldsymbol{d}^{(n)}\rangle\big\} = 0. \tag{11.5.4}$$

MIRT *See* `pgd_step.m`.

11.5.3 Asymptotic convergence rate

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

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

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

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

By this construction, it follows that

e,opt,psd,norm1

$$\big\|\boldsymbol{x}^{(n+1)} - \hat{\boldsymbol{x}}\big\|_{\hat{\boldsymbol{H}}^{1/2}} = \min_{\alpha}\Big\|\boldsymbol{x}^{(n)} + \alpha\hat{\boldsymbol{d}}^{(n)} - \hat{\boldsymbol{x}}\Big\|_{\hat{\boldsymbol{H}}^{1/2}} = \min_{\alpha}\Big\|\Big(\boldsymbol{I} - \alpha\boldsymbol{P}\hat{\boldsymbol{H}}\Big)(\boldsymbol{x}^{(n)} - \hat{\boldsymbol{x}})\Big\|_{\hat{\boldsymbol{H}}^{1/2}}. \tag{11.5.5}$$

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

$$\big\|\boldsymbol{\delta}^{(n+1)}\big\|_{\tilde{\boldsymbol{H}}^{1/2}} = \min_{\alpha}\Big\|(\boldsymbol{I} - \alpha\tilde{\boldsymbol{H}})\boldsymbol{\delta}^{(n)}\Big\|_{\tilde{\boldsymbol{H}}^{1/2}} \leq \min_{\alpha}\Big\|\boldsymbol{I} - \alpha\tilde{\boldsymbol{H}}\Big\|_{\tilde{\boldsymbol{H}}^{1/2}} \|\boldsymbol{\delta}^{(n)}\|_{\tilde{\boldsymbol{H}}^{1/2}}.$$

In particular, following [27, p. 31],

$$\Big\|\boldsymbol{I} - \alpha\tilde{\boldsymbol{H}}\Big\|_{\tilde{\boldsymbol{H}}^{1/2}} = \arg\max_{\boldsymbol{x}\neq\boldsymbol{0}} \sqrt{\frac{\boldsymbol{x}'(\boldsymbol{I} - \alpha\tilde{\boldsymbol{H}})\tilde{\boldsymbol{H}}(\boldsymbol{I} - \alpha\tilde{\boldsymbol{H}})\boldsymbol{x}}{\|\boldsymbol{x}\|^2}} = \arg\max_{j=1,\ldots,n_{\mathrm{p}}} |1 - \alpha\lambda_j|\sqrt{\lambda_j},$$

where the λ_j values are the eigenvalues of $\tilde{\boldsymbol{H}}$. Thus

e,opt,psd,cond

$$\min_{\alpha}\Big\|\boldsymbol{I} - \alpha\tilde{\boldsymbol{H}}\Big\|_{\tilde{\boldsymbol{H}}^{1/2}} = \min_{\alpha}\arg\max_{j=1,\ldots,n_{\mathrm{p}}} |1 - \alpha\lambda_j|\sqrt{\lambda_j} = \frac{\kappa - 1}{\kappa + 1}, \tag{11.5.6}$$

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

e,opt,psd,norm

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

So the closer κ is to unity, the faster the convergence should be. By (27.1.1): $\kappa = \kappa\left(\tilde{\boldsymbol{H}}\right) = \kappa\left(\boldsymbol{P}^{1/2}\hat{\boldsymbol{H}}\boldsymbol{P}^{1/2}\right) = \kappa\left(\boldsymbol{P}\hat{\boldsymbol{H}}\right)$, so one would like to choose $\boldsymbol{P}$ to minimize the condition number of the product $\boldsymbol{P}\hat{\boldsymbol{H}}$.

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

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

See [28] for discussion of choosing step sizes to improve the worst-case convergence rate, and see [29] for worst-case analysis of **strongly convex** problems.

11.6 Line search methods (s,opt,line)

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** [30] and the **bisection method** (or **binary search**). See also [31] [32]. A monotonic surrogate-function method well-suited to inverse problems is described in §14.5.6. Rarely is the minimization in (11.5.1) exact. Often one seeks α_n that satisfies the **Wolfe conditions** [33, 34], including the **Armijo rule** [35]:

$$\Psi(\boldsymbol{x}^{(n)} + \alpha_n \boldsymbol{d}^{(n)}) \leq \Psi(\boldsymbol{x}^{(n)}) + c_1 \alpha_n \operatorname{real}\{\langle \nabla\Psi(\boldsymbol{x}^{(n)}),\, \boldsymbol{d}^{(n)}\rangle\}$$

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 Ψ to help accelerate the line search.

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

s,opt,line,lips

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

$$\begin{aligned}
\left|\dot{f}(a) - \dot{f}(b)\right| &= |\operatorname{real}\{\langle \nabla\Psi(\boldsymbol{x} + a\boldsymbol{d}) - \nabla\Psi(\boldsymbol{x} + b\boldsymbol{d}),\, \boldsymbol{d}\rangle\}| \\
&\leq \left|\langle \boldsymbol{S}^{-1}\left(\nabla\Psi(\boldsymbol{x} + a\boldsymbol{d}) - \nabla\Psi(\boldsymbol{x} + b\boldsymbol{d})\right),\, \boldsymbol{S}'\boldsymbol{d}\rangle\right| \\
&\leq \left\|\boldsymbol{S}^{-1}\left(\nabla\Psi(\boldsymbol{x} + a\boldsymbol{d}) - \nabla\Psi(\boldsymbol{x} + b\boldsymbol{d})\right)\right\| \|\boldsymbol{S}'\boldsymbol{d}\| \\
&\leq \|\boldsymbol{S}'\left(a\boldsymbol{d} - b\boldsymbol{d}\right)\| \leq \|\boldsymbol{S}'\boldsymbol{d}\|^2 |a - b| .
\end{aligned}$$

Thus $\dot{f}$ has Lipschitz constant $\mathcal{L}_{\dot{f}} = \|\boldsymbol{S}'\boldsymbol{d}\|^2 = \boldsymbol{d}'\boldsymbol{S}\boldsymbol{S}'\boldsymbol{d}$, and we can apply **gradient descent** to f:

$$\alpha^{(k+1)} = \alpha^{(k)} - \frac{1}{\mathcal{L}_{\dot{f}}}\dot{f}(\alpha^{(k)}) = \alpha^{(k)} - \frac{1}{\mathcal{L}_{\dot{f}}}\operatorname{real}\left\{\langle \nabla\Psi\left(\boldsymbol{x} + \alpha^{(k)}\boldsymbol{d}\right),\, \boldsymbol{d}\rangle\right\}.$$

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)

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

$$\psi(\alpha) = \Psi(\boldsymbol{x}^{(n)} + \alpha\boldsymbol{d}^{(n)}) .$$

Then by (11.5.3) or (11.5.4):

$$\dot{\psi}(\alpha) = \operatorname{real}\{\langle \nabla\Psi(\boldsymbol{x}^{(n)} + \alpha\boldsymbol{d}^{(n)}),\, \boldsymbol{d}^{(n)}\rangle\}$$

and[7]

$$\ddot{\psi}(\alpha) = \langle \boldsymbol{d}^{(n)},\, \nabla^2\Psi(\boldsymbol{x}^{(n)} + \alpha\boldsymbol{d}^{(n)})\, \boldsymbol{d}^{(n)}\rangle \qquad (11.6.1)$$

[7] In complex case, we assume $\nabla\Psi$ is holomorphic and define $\nabla^2\Psi$ as in §29.4.

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

e,opt,psd,ddot,f,approx

$$= \frac{1}{\epsilon}\left(\text{real}\{\langle \nabla\Psi(\boldsymbol{x}^{(n)} + (\alpha+\epsilon)\boldsymbol{d}^{(n)}),\ \boldsymbol{d}^{(n)}\rangle\} - \text{real}\{\langle \nabla\Psi(\boldsymbol{x}^{(n)} + \alpha\boldsymbol{d}^{(n)}),\ \boldsymbol{d}^{(n)}\rangle\}\right). \tag{11.6.3}$$

Thus the approximate minimizer is

$$\alpha_n = -\frac{\dot{\psi}(0)}{\ddot{\psi}(0)} \approx \frac{-\text{real}\{\langle \nabla\Psi(\boldsymbol{x}^{(n)}),\ \boldsymbol{d}^{(n)}\rangle\}}{\frac{1}{\epsilon}\text{real}\{\langle \nabla\Psi(\boldsymbol{x}^{(n)} + \epsilon\boldsymbol{d}^{(n)}) - \nabla\Psi(\boldsymbol{x}^{(n)}),\ \boldsymbol{d}^{(n)}\rangle\}}.$$

This approximation requires one extra evaluation of the cost function gradient $\nabla\Psi$ and avoids computing the Hessian of Ψ. 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)

s,opt,line,pwls

For general cost functions, repeatedly evaluating Ψ 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 Ψ and its gradient for a line search fairly efficiently. If

$$\Psi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \beta \sum_k \psi([\boldsymbol{C}\boldsymbol{x}]_k),$$

then we can write the line search cost function as

$$f(\alpha) = \Psi(\boldsymbol{x}^{(n)} + \alpha\boldsymbol{d}^{(n)}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\left(\boldsymbol{x}^{(n)} + \alpha\boldsymbol{d}^{(n)}\right)\|_2^2 + \beta\sum_k \psi([\boldsymbol{C}\left(\boldsymbol{x}^{(n)} + \alpha\boldsymbol{d}^{(n)}\right)]_k) \tag{11.6.4}$$

e,opt,line,eff

$$= \frac{1}{2}\|\boldsymbol{y} - \left(\boldsymbol{A}\boldsymbol{x}^{(n)}\right) - \alpha\left(\boldsymbol{A}\boldsymbol{d}^{(n)}\right)\|_2^2 + \beta\sum_k \psi([\boldsymbol{C}\boldsymbol{x}^{(n)}]_k + \alpha[\boldsymbol{C}\boldsymbol{d}^{(n)}]_k). \tag{11.6.5}$$

We can precompute the products $\boldsymbol{A}\boldsymbol{x}^{(n)}$, $\boldsymbol{A}\boldsymbol{d}^{(n)}$, $\boldsymbol{C}\boldsymbol{x}^{(n)}$ and $\boldsymbol{C}\boldsymbol{d}^{(n)}$ before starting the line search, and then use (11.6.5). Note that

$$\begin{aligned}\boldsymbol{A}\boldsymbol{x}^{(n+1)} &= \boldsymbol{A}\left(\boldsymbol{x}^{(n)} + \alpha_n\boldsymbol{d}^{(n)}\right)\\ &= \left(\boldsymbol{A}\boldsymbol{x}^{(n)}\right) + \alpha_n\left(\boldsymbol{A}\boldsymbol{d}^{(n)}\right),\end{aligned}$$

so we can determine $\boldsymbol{A}\boldsymbol{x}^{(n+1)}$ from the vectors $\boldsymbol{A}\boldsymbol{x}^{(n)}$ and $\boldsymbol{A}\boldsymbol{d}^{(n)}$ computed before starting the line search.

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

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 Ψ 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 [36].

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

s,opt,bb

The **Barzilai and Borwein gradient method** (**BBGM**) for large-scale optimization problems is the following simple iteration [37–40]:

$$\boxed{\begin{aligned}\boldsymbol{g}^{(n)} &\triangleq \nabla\Psi(\boldsymbol{x}^{(n)})\\ \alpha_n &= \frac{\left\|\boldsymbol{x}^{(n)} - \boldsymbol{x}^{(n-1)}\right\|^2}{\langle \boldsymbol{x}^{(n)} - \boldsymbol{x}^{(n-1)},\ \boldsymbol{g}^{(n)} - \boldsymbol{g}^{(n-1)}\rangle}\\ \boldsymbol{x}^{(n+1)} &= \boldsymbol{x}^{(n)} - \alpha_n\,\boldsymbol{g}^{(n)}.\end{aligned}}$$

This method has global convergence when Ψ is a strictly convex quadratic, even though it is not guaranteed to decrease the cost function monotonically [38, 41].

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 [39] proposed a modification that adjusts the step size α_n to enforce the following condition:

$$\Psi\left(\boldsymbol{x}^{(n+1)}\right) \leq \max_{0 \leq j \leq M} \Psi\left(\boldsymbol{x}^{(n-j)}\right) + \gamma \left\langle \boldsymbol{x}^{(n+1)} - \boldsymbol{x}^{(n)},\, \boldsymbol{g}^{(n)} \right\rangle,$$

where M is a nonnegative integer and γ is a small positive number, following [42, 43].

Using §11.3.8, one can derive the following preconditioned version. (Problem 11.8.)

$$\boxed{\begin{gathered} \boldsymbol{g}^{(n)} \triangleq \nabla \Psi(\boldsymbol{x}^{(n)}) \\ \alpha_n = \frac{\alpha_{n-1}^2 \left\langle \boldsymbol{g}^{(n-1)},\, \boldsymbol{P} \boldsymbol{g}^{(n-1)} \right\rangle}{\left\langle \boldsymbol{x}^{(n)} - \boldsymbol{x}^{(n-1)},\, \boldsymbol{g}^{(n)} - \boldsymbol{g}^{(n-1)} \right\rangle} \\ \boldsymbol{x}^{(n+1)} = \boldsymbol{x}^{(n)} - \alpha_n \boldsymbol{P} \boldsymbol{g}^{(n)}. \end{gathered}}$$

x,opt,bb

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 α_0. Clearly the iterates can be non-monotonic.*

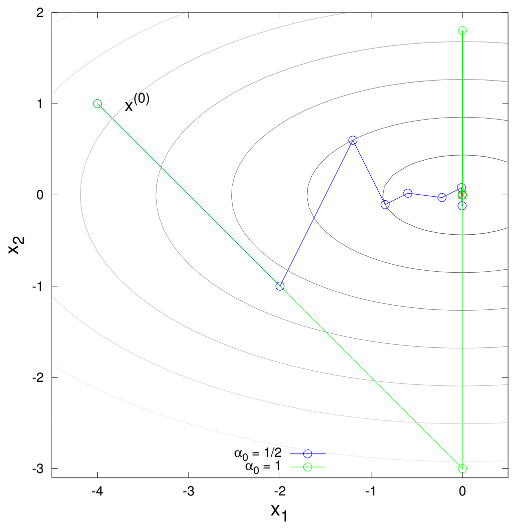


Figure 11.7.1: Fast convergence of BBGM for the same example shown in Fig. 11.3.1. In this (strongly convex, quadratic) case, BBGM with $\alpha_0 = 1$ (green) converges exactly in 3 iterations, and converges rapidly with $\alpha_0 = 1/2$ (blue). fig_opt_bb1

MIRT *See* `qpwls_bb1.m`.

See [44] for other step-size rules.

s,opt,nesterov

11.7.2 Nesterov's "optimal" first-order methods (s,opt,nesterov)

Nesterov [45, 46] 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 $\boldsymbol{S}$-Lipschitz continuous per (11.3.2) (*cf.* [47–49]), leading to the following algorithm. Starting with $\boldsymbol{z}^{(0)} = \boldsymbol{x}^{(0)}$ and $t_0 = 1$, Nesterov's **fast gradient method** (**FGM**) is:

$$\boxed{\begin{aligned} t_{n+1} &= \left(1+\sqrt{1+4t_n^2}\right)/2 \\ \boldsymbol{x}^{(n+1)} &= \boldsymbol{z}^{(n)} - [\boldsymbol{S}\boldsymbol{S}']^{-1}\,\nabla\Psi(\boldsymbol{z}^{(n)}) \\ &= \arg\min_{\boldsymbol{x}} \phi^{(n)}(\boldsymbol{x}), \quad \phi^{(n)}(\boldsymbol{x}) \triangleq \Psi(\boldsymbol{z}^{(n)}) + \nabla\Psi(\boldsymbol{z}^{(n)})\,(\boldsymbol{x}-\boldsymbol{z}^{(n)}) + \frac{1}{2}\left\|\boldsymbol{x}-\boldsymbol{z}^{(n)}\right\|_{\boldsymbol{S}\boldsymbol{S}'}^2 \\ \boldsymbol{z}^{(n+1)} &= \boldsymbol{x}^{(n+1)} + \frac{t_n - 1}{t_{n+1}}\left(\boldsymbol{x}^{(n+1)} - \boldsymbol{x}^{(n)}\right). \end{aligned}}$$

The ratio $\frac{t_n-1}{t_{n+1}}$ for the momentum term approaches 2 as n increases. Because $t_0 - 1 = 0$, the first iteration is simply PGD with preconditioner $\boldsymbol{P} = [\boldsymbol{S}\boldsymbol{S}']^{-1}$.

This method reaches an "ε optimal solution" where $\Psi(\boldsymbol{x}^{(n)}) \leq \Psi(\hat{\boldsymbol{x}})+\varepsilon$ within $O(1/\sqrt{\epsilon})$ iterations, whereas conventional PGD requires $O(1/\epsilon)$ iterations [45, 46]. Specifically, if Ψ is convex and has Lipschitz gradient, then

$$\Psi(\boldsymbol{x}^{(n)}) - \Psi(\hat{\boldsymbol{x}}) \leq \frac{2\left\|\boldsymbol{S}'\left(\boldsymbol{x}^{(0)}-\hat{\boldsymbol{x}}\right)\right\|^2}{n^2}. \tag{11.7.1}$$

e,opt,nesterov,1/n2

This $O(1/n^2)$ convergence rate is optimal in a certain (global, non-asymptotic) sense [5, p. 77].

The above Nesterov method requires a Lipschitz constant $\mathcal{L}$ or matrix $\boldsymbol{S}$, whereas PSD, BBGM, and PCG do not. There are line-search variants, *e.g.*, [50], that do not require $\mathcal{L}$. See [51] 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 $\boldsymbol{S} = \sqrt{\mathcal{L}}\boldsymbol{I}$.

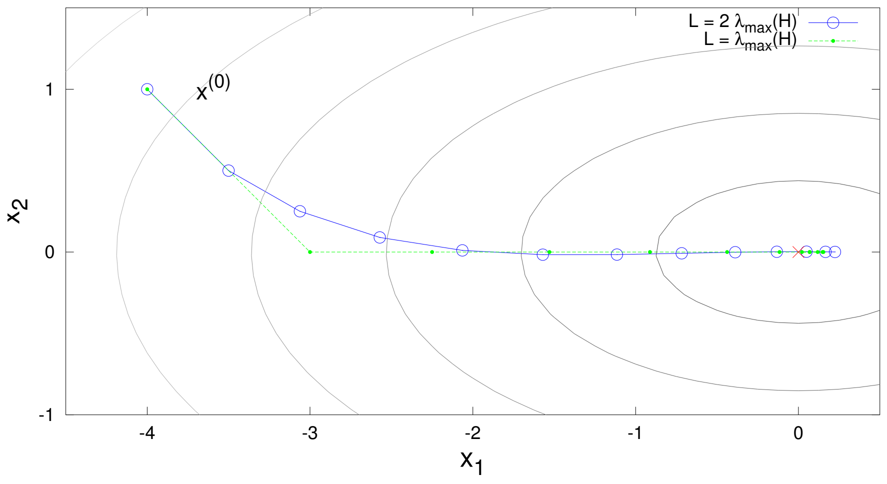


Figure 11.7.2: Illustration of Nesterov method for the same example shown in Fig. 11.3.1. For the blue curve, $\mathcal{L} = 2\rho(\nabla^2\Psi)$, reflecting the fact that often the maximum eigenvalue is over-estimated. For the green curve, $\mathcal{L} = \rho(\nabla^2\Psi)$, which is the ideal Lipschitz constant. In this (strongly convex, quadratic) case, the momentum term causes the iterates to slightly overshoot the minimizer.

fig_opt_nest1

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 [5, Thm 2.1.7] [21].

The **optimized gradient method** (**OGM**) [52] tightens the bound (11.7.1) by a factor of two using a similarly efficient algorithm. By constructing a lower bound, Drori showed that OGM is an optimal first-order method [53].

Momentum is also used for training **artificial neural nets** via (stochastic) gradient descent [54, 55].

s,opt,pcg

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.

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 Ψ. This approach ensures a more efficient search over the parameter space, and even leads to convergence in n_{p} iterations [27, p. 36] for quadratic cost functions (in the absence of numerical roundoff errors).

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

e,opt,pcg,conj

$$\left\langle \boldsymbol{d}^{(n)},\, \boldsymbol{H}_{n-1} \boldsymbol{d}^{(n-1)} \right\rangle = 0. \tag{11.8.1}$$

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 $\boldsymbol{d}^{(n)}$ using the recursion[8]

e,opt,pcg,dir

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

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

e,opt,pcg,gamn,daniel

$$\gamma_n^{\mathrm{D}} \triangleq \frac{\left\langle \boldsymbol{H}_{n-1} \boldsymbol{d}^{(n-1)},\, \boldsymbol{P}\boldsymbol{g}^{(n)} \right\rangle}{\left\langle \boldsymbol{H}_{n-1} \boldsymbol{d}^{(n-1)},\, \boldsymbol{d}^{(n-1)} \right\rangle}. \tag{11.8.3}$$

Proposed by Daniel [56], this choice for γ_n satisfies the conjugacy condition (11.8.1) exactly, but is inconvenient (for nonquadratic cost functions) because it depends on the Hessian $\boldsymbol{H}_{n-1}$. Furthermore, after multiple iterations of the recursion (11.8.3), it is possible that the search direction $\boldsymbol{d}^{(n)}$ will not be a descent direction. Several alternative choices for γ_n have been proposed that overcome these disadvantages. Historically [57], 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:

e,opt,pcg,gn,gnp

$$\begin{aligned} \boldsymbol{g}^{(n)} &\approx \boldsymbol{g}^{(n-1)} + \boldsymbol{H}_{n-1} \left(\boldsymbol{x}^{(n)} - \boldsymbol{x}^{(n-1)} \right) & (11.8.4) \\ &= \boldsymbol{g}^{(n-1)} + \alpha_{n-1} \boldsymbol{H}_{n-1} \boldsymbol{d}^{(n-1)}, & (11.8.5) \end{aligned}$$

exploiting (11.5.2), *i.e.*, $\boldsymbol{H}_{n-1} \boldsymbol{d}^{(n-1)} = (\boldsymbol{g}^{(n)} - \boldsymbol{g}^{(n-1)}) / \alpha_{n-1}$. (This relation is exact for quadratic cost functions.) Substituting into (11.8.3) yields the **Hestenes-Stiefel** formula [58]:

e,opt,pcg,gamn,hs

$$\gamma_n^{\mathrm{HS}} \triangleq \frac{\left\langle \boldsymbol{g}^{(n)} - \boldsymbol{g}^{(n-1)},\, \boldsymbol{P}\boldsymbol{g}^{(n)} \right\rangle}{\left\langle \boldsymbol{g}^{(n)} - \boldsymbol{g}^{(n-1)},\, \boldsymbol{d}^{(n-1)} \right\rangle}. \tag{11.8.6}$$

For further variations, invoke the orthogonality condition (11.5.3) or (11.5.4), *i.e.*, $0 = \mathrm{real}\left\{ \left\langle \boldsymbol{g}^{(n)},\, \boldsymbol{d}^{(n-1)} \right\rangle \right\}$, 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:

$$\left\langle \boldsymbol{g}^{(n)} - \boldsymbol{g}^{(n-1)},\, \boldsymbol{d}^{(n-1)} \right\rangle = -\left\langle \boldsymbol{g}^{(n-1)},\, \boldsymbol{d}^{(n-1)} \right\rangle = -\left\langle \boldsymbol{g}^{(n-1)},\, -\boldsymbol{P}\boldsymbol{g}^{(n-1)} + \gamma_{n-1} \boldsymbol{d}^{(n-2)} \right\rangle = \left\langle \boldsymbol{g}^{(n-1)},\, \boldsymbol{P}\boldsymbol{g}^{(n-1)} \right\rangle.$$

Substituting into (11.8.6) yields the **Polak-Ribiere** [59, 60] choice:

e,opt,pcg,gamn,pr

$$\gamma_n^{\mathrm{PR}} \triangleq \frac{\left\langle \boldsymbol{g}^{(n)} - \boldsymbol{g}^{(n-1)},\, \boldsymbol{P}\boldsymbol{g}^{(n)} \right\rangle}{\left\langle \boldsymbol{g}^{(n-1)},\, \boldsymbol{P}\boldsymbol{g}^{(n-1)} \right\rangle}. \tag{11.8.7}$$

This choice has been used frequently for image reconstruction [31, 61, 62].

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

$$\begin{aligned} -\left\langle \boldsymbol{P}\boldsymbol{g}^{(n-1)},\, \boldsymbol{g}^{(n)} \right\rangle = \left\langle \boldsymbol{d}^{(n-1)} - \gamma_{n-1} \boldsymbol{d}^{(n-2)},\, \boldsymbol{g}^{(n)} \right\rangle &= -\gamma_{n-1} \left\langle \boldsymbol{d}^{(n-2)},\, \boldsymbol{g}^{(n)} \right\rangle & (11.8.8) \\ &\approx -\gamma_{n-1} \left\langle \boldsymbol{d}^{(n-2)},\, \boldsymbol{g}^{(n-1)} + \alpha_{n-1} \boldsymbol{H}_{n-1} \boldsymbol{d}^{(n-1)} \right\rangle & (11.8.9) \end{aligned}$$

[8]Provided that $\boldsymbol{P}$ is a Hermitian postive definite **preconditioning matrix**, one can motivate (11.8.2) by working in the transformed coordinate system discussed in §11.3.8.

e,opt,pcg,pgg,0

$$= \gamma_{n-1}\alpha_{n-1} \langle \boldsymbol{d}^{(n-2)}, \boldsymbol{H}_{n-1}\boldsymbol{d}^{(n-1)} \rangle \approx 0, \tag{11.8.10}$$

where the last approximation follows from the conjugacy condition (11.8.1) *provided* that $\boldsymbol{H}_{n-1} \approx \boldsymbol{H}_{n-2}$. Applying (11.8.10) to (11.8.6) yields the formula of Dai and Yuan [63]:

e,opt,pcg,gamn,dy

$$\gamma_n^{\text{DY}} \triangleq \frac{\langle \boldsymbol{g}^{(n)}, \boldsymbol{P}\,\boldsymbol{g}^{(n)} \rangle}{\langle \boldsymbol{g}^{(n)} - \boldsymbol{g}^{(n-1)}, \boldsymbol{d}^{(n-1)} \rangle}. \tag{11.8.11}$$

Alternatively, applying (11.8.10) to (11.8.7) yields the classic **Fletcher-Reeves** formula [64]:

e,opt,pcg,gamn,fr

$$\gamma_n^{\text{FR}} \triangleq \frac{\langle \boldsymbol{g}^{(n)}, \boldsymbol{P}\,\boldsymbol{g}^{(n)} \rangle}{\langle \boldsymbol{g}^{(n-1)}, \boldsymbol{P}\,\boldsymbol{g}^{(n-1)} \rangle}. \tag{11.8.12}$$

Because $0 = \langle \boldsymbol{g}^{(n)}, \boldsymbol{d}^{(n-1)} \rangle$ for an exact line search, $\langle \boldsymbol{P}\left(\boldsymbol{g}^{(n)} - \boldsymbol{g}^{(n-1)}\right) - \zeta_n \boldsymbol{d}^{(n-1)}, \boldsymbol{g}^{(n)} \rangle$ is an alternative to the numerator of (11.8.6) for any value of ζ_n. Hager and Zhang [32, 65, 66] surveyed numerous choices for γ_n and proposed the following choice that has favorable convergence properties:

e,opt,pcg,gamn,hz

$$\gamma_n^{\text{HZ}} \triangleq \frac{\left\langle \boldsymbol{P}\left(\boldsymbol{g}^{(n)} - \boldsymbol{g}^{(n-1)}\right) - \zeta_n \boldsymbol{d}^{(n-1)}, \boldsymbol{g}^{(n)} \right\rangle}{\langle \boldsymbol{g}^{(n)} - \boldsymbol{g}^{(n-1)}, \boldsymbol{d}^{(n-1)} \rangle}, \qquad \zeta_n \triangleq 2\frac{\left\langle \boldsymbol{g}^{(n)} - \boldsymbol{g}^{(n-1)}, \boldsymbol{P}\left(\boldsymbol{g}^{(n)} - \boldsymbol{g}^{(n-1)}\right) \right\rangle}{\langle \boldsymbol{g}^{(n)} - \boldsymbol{g}^{(n-1)}, \boldsymbol{d}^{(n-1)} \rangle}. \tag{11.8.13}$$

(Problem 11.9.) This choice ensures monotonicity under mild conditions, unlike many of the previous alternatives. See also [23, 67–77].

In practice, often one uses $[\gamma_n]_+$ in place of γ_n for inexact line searches [32, 78]. For cases where $\boldsymbol{x}$ is complex, see [79–81].

The following summarizes the PCG algorithm for one of the many options for γ_n.

PCG Algorithm (Polak-Ribiere version of γ_n)

$$\begin{aligned}
\boldsymbol{g}^{(n)} &= \nabla \Psi(\boldsymbol{x}^{(n)}) && \text{(gradient)} \\
\boldsymbol{p}^{(n)} &= \boldsymbol{P}\,\boldsymbol{g}^{(n)} && \text{(precondition)} \\
\gamma_n &= \begin{cases} 0, & n = 0 \\ \dfrac{\text{real}\{\langle \boldsymbol{g}^{(n)} - \boldsymbol{g}^{(n-1)}, \boldsymbol{p}^{(n)} \rangle\}}{\text{real}\{\langle \boldsymbol{g}^{(n-1)}, \boldsymbol{p}^{(n-1)} \rangle\}}, & n > 0 \end{cases} && \text{(Polak-Ribiere)} \\
\boldsymbol{d}^{(n)} &= -\boldsymbol{p}^{(n)} + \gamma_n \boldsymbol{d}^{(n-1)} && \text{(search direction)} \quad (11.8.14) \\
\alpha_n &= \mathop{\arg\min}_{\alpha \in [0,\infty)} \Psi(\boldsymbol{x}^{(n)} + \alpha \boldsymbol{d}^{(n)}) && \text{(step size)} \quad (11.8.15) \\
\boldsymbol{x}^{(n+1)} &= \boldsymbol{x}^{(n)} + \alpha_n \boldsymbol{d}^{(n)} && \text{(update)} \quad (11.8.16)
\end{aligned}$$

e,opt,pcg,dirn
e,opt,pcg,step
e,opt,pcg,alg

For nonquadratic cost functions Ψ, one must find α_n using a line-search. §11.6 and §14.5.6 describe efficient line-search methods suitable for many imaging problems, based on **majorize-minimize** methods to determine the step sizes [62] [82].

Generalizations of PCG using multiple search directions and/or preconditioners have also been proposed [83–86]. A related generalization is subspace minimization using multiple search directions simultaneously [87].

s,opt,pcg,rate

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 [27, p. 51]:

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

where κ is the **condition number** of the preconditioned Hessian $\boldsymbol{P}^{1/2}\boldsymbol{H}\boldsymbol{P}^{1/2}$ corresponding to the cost function $\Psi\left(\boldsymbol{P}^{1/2}\cdot\right)$. 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 κ). Even tighter error bounds as a function of iteration, involving nth-order polynomials with roots near the eigenvalues of the (preconditioned) Hessian, are given in [27, p. 51]. In short, PCG converges **quadratically** [5, p. 45].

Recently, nonlinear PCG algorithms have been developed that are globally convergent under certain conditions on the line search [32, 65, 66].

e,opt,alg,qn

$$\begin{aligned}
\boldsymbol{x}^{(n+1)} &= \boldsymbol{x}^{(n)} - \boldsymbol{P}_n \nabla\Psi(\boldsymbol{x}^{(n)}) && (11.9.1)\\
\boldsymbol{\delta}^{(n)} &\triangleq \boldsymbol{x}^{(n+1)} - \boldsymbol{x}^{(n)}\\
\boldsymbol{q}^{(n)} &\triangleq \nabla\Psi\big(\boldsymbol{x}^{(n+1)}\big) - \nabla\Psi(\boldsymbol{x}^{(n)})\\
\boldsymbol{w}^{(n)} &\triangleq \boldsymbol{\delta}^{(n)} - \boldsymbol{P}_n \boldsymbol{q}^{(n)}\\
b_n &\triangleq \frac{1}{\langle \boldsymbol{w}^{(n)},\, \boldsymbol{q}^{(n)} \rangle}\\
\boldsymbol{P}_{n+1} &= \boldsymbol{P}_n + b_n \boldsymbol{w}^{(n)} \left[\boldsymbol{w}^{(n)}\right]'. && (11.9.2)
\end{aligned}$$

e,opt,qn,P

tab,qn

Table 11.1: Quasi-Newton algorithm.

11.9 Quasi-Newton methods (s,opt,qn)

s,opt,qn

The preconditioning matrix $\boldsymbol{P}$ remains unchanged throughout the iterations of the preceding algorithms, except in the Newton-Raphson iteration (11.4.3), in which $\boldsymbol{P}$ 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 *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 [88] to adjust the preconditioner each iteration, see [22, p. 136] and [89].

A **quasi-Newton** algorithm begins with some initial guess $\boldsymbol{P}_0$ of the inverse Hessian $\boldsymbol{H}^{-1}(\hat{\boldsymbol{x}})$, and updates both $\boldsymbol{x}^{(n)}$ *and* $\boldsymbol{P}_n$ each iteration. The principle behind quasi-Newton methods is the **secant condition**, which originates from the gradient of the quadratic approximation (11.4.1) evaluated at $\boldsymbol{x}^{(n+1)}$:

$$\nabla\Psi\big(\boldsymbol{x}^{(n+1)}\big) \approx \nabla\Psi(\boldsymbol{x}^{(n)}) + \boldsymbol{H}_n(\boldsymbol{x}^{(n+1)} - \boldsymbol{x}^{(n)}).$$

Defining

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

and

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

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

$$\boldsymbol{P}_{n+1}\boldsymbol{q}^{(n)} = \boldsymbol{\delta}^{(n)}.$$

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

$$\boldsymbol{P}_{n+1} = \boldsymbol{P}_n + b_n \boldsymbol{w}^{(n)} \left[\boldsymbol{w}^{(n)}\right]'.$$

Substituting into the (inverse) secant condition we see that

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

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

$$\boldsymbol{w}^{(n)} = \boldsymbol{\delta}^{(n)} - \boldsymbol{P}_n \boldsymbol{q}^{(n)},$$

in which case we must have

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

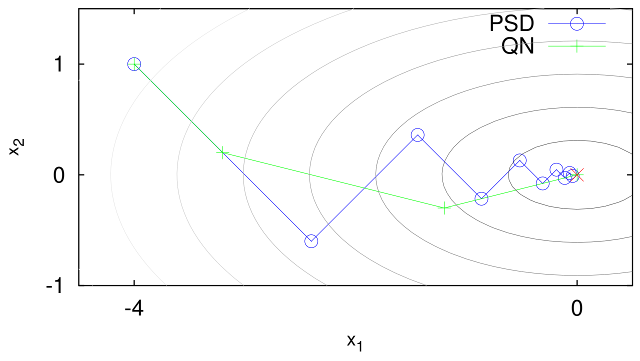
Combining these formulas leads to the **quasi-Newton** algorithm shown in Table 11.1, also known as **Broyden's method** [2, p. 77]. One can incorporate a line-search [90] or step-halving procedure into (11.9.1).

Convergence conditions are discussed in [91, 92] and [2, p. 77]. The asymptotic convergence rates of QN can be cumbersome to analyze [2, p. 78]. See [92] for cases where QN algorithms converge Q-superlinearly.

Mat *The* MATLAB *function* `fminunc` *is based on a quasi-Newton method.*

x,opt,qn

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 $\boldsymbol{P}_0 = \frac{1}{5}\boldsymbol{I}$. Because the cost function is quadratic, and because $n_{\mathrm{p}} = 2$, after two rank-one updates the inverse Hessian $\boldsymbol{P}_2$ is exactly $\boldsymbol{H}^{-1}$, so $\boldsymbol{x}^{(3)} = \hat{\boldsymbol{x}}$,* i.e., *convergence in $n_{\mathrm{p}} + 1$ iterations as expected theoretically [22, p. 138].*



fig_qn1

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 $\boldsymbol{P}_n$ in the algorithm is the operation of multiplying by vectors in $\mathbb{R}^{n_\mathrm{p}}$, which one can implement efficiently as follows:

e,opt,qn,Px

$$\boldsymbol{P}_n \boldsymbol{x} = \left[\boldsymbol{P}_0 + \sum_{i=1}^{n-1} b_i \boldsymbol{w}^{(i)} \left[\boldsymbol{w}^{(i)}\right]'\right] \boldsymbol{x} = \boldsymbol{P}_0 \boldsymbol{x} + \sum_{i=1}^{n-1} b_i \left\langle \boldsymbol{w}^{(i)},\, \boldsymbol{x} \right\rangle \boldsymbol{w}^{(i)}. \tag{11.9.3}$$

As long as $n \ll n_\mathrm{p}$, 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-Shanno** (**BFGS**) method, for which limited-memory versions are available, *e.g.*, [93]. 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 $\boldsymbol{w}^{(i)}$ vectors. A second reason is that for nonquadratic cost functions, the quadratic approximation (11.4.1) may be poor for the initial $\boldsymbol{x}^{(n)}$ values, so "forgetting" the effects of those $\boldsymbol{x}^{(n)}$ values as n decreases may be helpful.

Another subtle practical consideration is the need to monitor whether $\boldsymbol{P}_n$ remains positive definite, because the b_n values can be negative [22, p. 137]. The BFGS method may mitigate this concern [1, p. 63].

Choice of the initial inverse Hessian approximation $\boldsymbol{P}_0$ is critical [22, 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 Ψ. This idea stems from Lange's proposal to use the EM complete-data information matrix [94].

For imaging applications, see [95–98].

s,opt,bcm

11.10 Block coordinate descent methods (s,opt,bcm)

All of the preceding algorithms in this *chapter* update all pixels (elements of $\boldsymbol{x}$) simultaneously. In many optimization problems it can be beneficial to update a *group* of parameters simultaneously, holding all other parameters at their most recent values. Such methods go by various names including **block coordinate minimization** (**BCM**), **block coordinate descent** (**BCD**), **grouped coordinate descent** (**GCD**), **alternating minimization** (**AM**), or simply **coordinate descent** (**CD**). The literature does not have standard terminology distinguishing these methods.

All such methods partition the optimization variable $\boldsymbol{x}$ into two or more "blocks" (shorter vectors) and update one block while holding the other blocks constant. Some BCD methods use a fixed partition, whereas others allow the partition to change as the iterations proceed.

We start with the simpler case where the partition is fixed at the outset. Partition the vector $\boldsymbol{x} \in \mathbb{R}^{n_\mathrm{p}}$ into K blocks:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_K \end{bmatrix} \in \mathbb{R}^{n_\mathrm{p}},$$

where $\boldsymbol{x}_k \in \mathbb{R}^{n_k}$ for $k = 1, \ldots, K$ and where $\sum_{k=1}^K n_k = n_\mathrm{p}$. Then rewrite $\Psi(\boldsymbol{x})$ as $\Psi(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K)$.

11.10.1 Block coordinate minimization (BCM)

The conceptually simplest approach sequentially minimizes over each block while holding the others fixed.

Block coordinate minimization (BCM) "Algorithm"

```
for $n = 0, 1, \dots$ {
    for $k = 1, \dots, K$ {
        $$\boldsymbol{x}_k^{(n+1)} := \arg\min_{\boldsymbol{x}_k \in \mathbb{R}^{n_k}} \Psi\left(\boldsymbol{x}_1^{(n+1)}, \dots, \boldsymbol{x}_{k-1}^{(n+1)}, \boldsymbol{x}_k, \boldsymbol{x}_{k+1}^{(n)}, \dots, \boldsymbol{x}_K^{(n)}\right). \quad (11.10.1)$$
    }
}
```

e,opt,bcm

Variants include choosing the block order non-sequentially, including random order selection.

11.10.2 Block coordinate descent (BCD)

In some applications it is impractical to solve the inner block minimization (11.10.1) exactly, and we settle for simply *decreasing* the cost function for each update, rather than finding each block-wise minimizer.

Block coordinate descent (BCD) "Algorithm"

```
for $n = 0, 1, \dots$ {
    for $k = 1, \dots, K$ {
        Find $\boldsymbol{x}_k^{(n+1)} \in \mathbb{R}^{n_k}$ such that $\Psi\left(\boldsymbol{x}_1^{(n+1)}, \dots, \boldsymbol{x}_{k-1}^{(n+1)}, \boldsymbol{x}_k^{(n+1)}, \boldsymbol{x}_{k+1}^{(n)}, \dots, \boldsymbol{x}_K^{(n)}\right)$
            $\leq \Psi\left(\boldsymbol{x}_1^{(n+1)}, \dots, \boldsymbol{x}_{k-1}^{(n+1)}, \boldsymbol{x}_k^{(n)}, \boldsymbol{x}_{k+1}^{(n)}, \dots, \boldsymbol{x}_K^{(n)}\right).$   (11.10.2)
    }
}
```

e,opt,bcd

The MM approaches in Chapter 14 can be particularly useful for such **inexact updates**.

When written side by side, the difference between BCM (11.10.1) and BCD (11.10.2) is clear, but the literature rarely carefully distinguishes the two forms.

11.10.3 Iteration-dependent partitions

It is conceptually straightforward to generalize BCM and BCD to cases where the block partition can change each iteration, but describing such methods mathematically requires extra notation.

An **index set** $\mathcal{S}$ is a nonempty subset of the set $\{1, \dots, n_\text{p}\}$. The set $\tilde{\mathcal{S}}$ denotes the complement of $\mathcal{S}$ in $\{1, \dots, n_\text{p}\}$. Let $|\mathcal{S}|$ denote the cardinality of $\mathcal{S}$. We use $\boldsymbol{x}_\mathcal{S}$ to denote the $|\mathcal{S}|$-dimensional vector consisting of the $|\mathcal{S}|$ elements of $\boldsymbol{x}$ indexed by the members of $\mathcal{S}$. We similarly define $\boldsymbol{x}_{\tilde{\mathcal{S}}}$ as the $n_\text{p} - |\mathcal{S}|$ dimensional vector consisting of the remaining elements of $\boldsymbol{x}$. For example, if $n_\text{p} = 5$ and $\mathcal{S} = \{1, 3, 4\}$, then $\tilde{\mathcal{S}} = \{2, 5\}$, $\boldsymbol{x}_\mathcal{S} = (x_1, x_3, x_4)$, and $\boldsymbol{x}_{\tilde{\mathcal{S}}} = (x_2, x_5)$. Occasionally we use $\mathcal{S}$ as a superscript of a function or matrix, to serve as a reminder that the function or matrix depends on the choice of $\mathcal{S}$.

One more notational convention is helpful. Functions like $\Psi(\boldsymbol{x})$ expect a n_p-dimensional vector argument, but it is often convenient to split the argument $\boldsymbol{x}$ into two vectors: $\boldsymbol{x}_\mathcal{S}$ and $\boldsymbol{x}_{\tilde{\mathcal{S}}}$, as defined above. Therefore, we define expressions such as the following to be equivalent: $\Psi(\boldsymbol{x}_\mathcal{S}, \boldsymbol{x}_{\tilde{\mathcal{S}}}) = \Psi(\boldsymbol{x})$.

Using that notation, in the general BCM method one sequences through different index sets $\mathcal{S}_n$ and updates only the elements $\boldsymbol{x}_{\mathcal{S}_n}$ of $\boldsymbol{x}$ while holding the other parameters $\boldsymbol{x}_{\tilde{\mathcal{S}}_n}$ fixed [31]. At the nth iteration one would like to assign $\boldsymbol{x}_{\mathcal{S}_n}^{(n+1)}$ to the argument that minimizes $\Psi\left(\boldsymbol{x}_{\mathcal{S}_n}, \boldsymbol{x}_{\tilde{\mathcal{S}}_n}^{(n)}\right)$ over $\boldsymbol{x}_{\mathcal{S}_n}$, as summarized in the following algorithm.

General BCM "Algorithm"

$$\text{Choose } \mathcal{S}_n$$
$$\boldsymbol{x}_{\tilde{\mathcal{S}}_n}^{(n+1)} = \boldsymbol{x}_{\tilde{\mathcal{S}}_n}^{(n)}$$
$$\boldsymbol{x}_{\mathcal{S}_n}^{(n+1)} = \mathop{\arg\min}_{\boldsymbol{x}_{\mathcal{S}_n} \in \mathbb{R}^{|\mathcal{S}_n|}} \Psi\left(\boldsymbol{x}_{\mathcal{S}_n}, \boldsymbol{x}_{\tilde{\mathcal{S}}_n}^{(n)}\right). \tag{11.10.3}$$

e,opt,bcm,direct

As always, the minimization will be subject to any applicable constraints. This type of approach has been applied in a variety of fields [99–102]. This approach is globally convergent under remarkably broad conditions [103].

In many imaging problems, the minimization (11.10.3) is difficult, even if $\mathcal{S}_n$ only consists of a few pixels. One could apply any of the previously described iterative methods, such as the Newton-Raphson algorithm, to perform the minimization (11.10.3), which would require subiterations within the iterations. Often one settles for descent, leading so a general BCD alternative to BCM, *e.g.*, by using an MM approach of Chapter 14 for updating $\boldsymbol{x}_{\mathcal{S}_n}$.

Specific examples of such methods in the imaging literature are numerous, including [104–108].

s,opt,bcm,conv 11.10.4 Convergence properties

Convergence results (for limit points) under weak assumptions are given in [3, p. 268]. See also [109–112]. Generalizations for non-smooth and non-convex functions are in [111, 113–116]. This is an evolving area because of growing interest in BCD methods. **Limitations of BCD** methods are difficulty with **parallelism** and getting stuck at non-stationary points for non-smooth cost functions.

s,opt,cd 11.10.5 Coordinate descent methods (s,opt,cd)

A special case of BCM/BCD is when each block is just x_j, a single pixel. Again one can minimize or descend, but typically both cases are called **coordinate descent** (**CD**) in the literature. This method is also called **nonlinear Gauss-Siedel** and the method of **alternating variables** [117, p. 53]. Here is the general method:

Coordinate Descent (CD) "Algorithm"

```
for $n = 0, 1, \ldots$ {
    for $j = 1, \ldots, n_{\mathrm{p}}$ {
        $x_j^{(n+1)} := \arg\min_{x_j} \Psi\left(x_1^{(n+1)}, \ldots, x_{j-1}^{(n+1)}, x_j, x_{j+1}^{(n)}, \ldots, x_{n_{\mathrm{p}}}^{(n)}\right).$   (11.10.4)
    }
}
```

e,opt,cd,alg

The operation in (11.10.4) 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 $\boldsymbol{x}$ are always used. An early use of such a method for tomography was in [118].

The coordinate descent approach is often characterized as "very inefficient" in the general optimization literature (*e.g.*, [119, p. 413]), and its computational requirements have been reported incorrectly in the image reconstruction literature (*e.g.*, [120]). Sauer and Bouman analyzed such algorithms using clever frequency domain arguments [121], 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 in tomography 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 §16.7.2), the computational requirements become (nearly) comparable with many other approaches if the system matrix is precomputed. However, there is a significant practical disadvantage that arises in very large inverse problems (*e.g.*, 3D PET and CT): 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.*, [122].

The general method described by (11.10.4) 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 (11.10.4), so we settle for methods that approximate the minimizer or at least monotonically decrease Ψ.

11.10.5.1 Coordinate-descent Newton (CD-NR)-Raphson

If $\Psi(\boldsymbol{x})$ were a quadratic functional, then the natural approach to performing the minimization in (11.10.4) would be Newton's method, which would yield the exact 1D minimizer in (11.10.4) in one iteration. When $\Psi(\boldsymbol{x})$ in (11.1.3) is nonquadratic, applying Newton's method to (11.10.4) does not guarantee monotonic decreases in Ψ, but one might still try it anyway and hope for the best. In practice, mild nonmonotonicity often does not seem to be a problem in some cases, as suggested by the success of this approach in [123, 124]. For such a **coordinate-descent Newton-Raphson** (CD-NR) algorithm, we replace (11.10.4) with the following update:

e,opt,alg,cd-nr

$$x_j^{(n+1)} = \left[x_j^{(n)} - \frac{\frac{\partial}{\partial x_j} \Psi(\boldsymbol{x})\Big|_{\boldsymbol{x}=\tilde{\boldsymbol{x}}}}{\frac{\partial^2}{\partial x_j^2} \Psi(\boldsymbol{x})\Big|_{\boldsymbol{x}=\tilde{\boldsymbol{x}}}} \right]_+ . \tag{11.10.5}$$

The optional $[\cdot]_+$ operation enforces the nonnegativity constraint, if needed.

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

11.10.5.2 Asymptotic convergence rate of CD

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

$$x_j^{(n+1)} \approx \left[x_j^{(n)} - \frac{\frac{\partial}{\partial x_j} \hat{\Psi}(\boldsymbol{x})\Big|_{\boldsymbol{x}=\tilde{\boldsymbol{x}}}}{\frac{\partial^2}{\partial x_j^2} \hat{\Psi}(\boldsymbol{x})\Big|_{\boldsymbol{x}=\hat{\boldsymbol{x}}}} \right]_+ .$$

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 §16.7.3. Specifically, from §16.7.3:

$$\left\| \boldsymbol{x}^{(n+1)} - \hat{\boldsymbol{x}} \right\|_{\boldsymbol{H}^{1/2}} \leq \rho(\boldsymbol{M}_{\mathrm{GS}}) \left\| \boldsymbol{x}^{(n)} - \hat{\boldsymbol{x}} \right\|_{\boldsymbol{H}^{1/2}},$$

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

Formal analysis of the convergence rate of "pure" CD (11.10.4) follows as a special case of the analysis of **SAGE** in [125], and agrees with the above approximation.

Convergence of the coordinate descent method for strictly convex, twice-differentiable cost functions is analyzed in detail in [126], including consideration of **box constraints**. Powell demonstrates that uniqueness of the "arg min" step is important to ensure convergence [127].

Convergence rate analysis, including constrained cases, is given in [115, 128].

s,opt,anneal

11.11 Annealing methods (s,opt,anneal)

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

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

However, in situations where Ψ 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 Ψ, 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.

11.11.1 Simulated annealing

The **simulated annealing** method was originally designed for optimization problems with discrete parameters, such as the famous traveling salesman problem [129]. 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.*, [130–132], or as an image post-processing procedure *e.g.*, [133, 134]. 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.*, [135–137], as well as to the ML emission tomography problem [138]. For these problems that have no local minimum, much better methods exist!

A concise summary of how one can adapt the **Metropolis algorithm** [139] to perform annealing is given in [119], 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**).

11.11.2 Deterministic annealing and graduated nonconvexity

Chapter 14 illustrates how to simplify the optimization problem of a difficult cost function Ψ by iteratively replacing it with a sequence of more easily minimized surrogate functions. One can apply a similar philosophy to multimodal cost functions Ψ, by replacing Ψ 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.*, [140–148].

11.11.3 Other global optimization methods

There is a rich literature on global optimization methods. For differentiable functions, one interesting approach is **multilevel coordinate search** [149] 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.

11.12 Problems (s,opt,prob)

s,opt,prob

p,opt,pgd,lipl

Problem 11.1 *Consider the differentiable cost function $\Psi(x) = \frac{1}{1+x^2}$. Determine the (smallest possible) Lipschitz constant $\mathcal{L}$ for its derivative $\dot{\Psi}$. What happens if we apply GD with $\alpha = 1/\mathcal{L}$?*

p,opt,pgd,mono

Problem 11.2 *Generalize the monotone convergence in norm theory of PGD in §11.3.2 to cases where $\boldsymbol{P}$ is not necessarily Hermitian positive definite.* *(Solve?)*

p,opt,pgd,circ

Problem 11.3 *The PGD method is applied to the quadratic cost function $\Psi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|^2 + \beta\frac{1}{2}\|\boldsymbol{C}\boldsymbol{x}\|^2$ where $\mathsf{F} = \boldsymbol{A}'\boldsymbol{A}$ and $\boldsymbol{R} = \boldsymbol{C}'\boldsymbol{C}$ are both circulant. The preconditioner is simply $\boldsymbol{P} = \boldsymbol{I}$ and suppose the optimal step size $\alpha_\star$ 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 $\boldsymbol{C} = \boldsymbol{I}$ and assume that F has a low-pass nature.*

p,opt,pgd,norm

Problem 11.4 *Prove the PGD convergence theorem Theorem 11.3.7.*

p,opt,pgd,1n,lem

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

p,opt,pgd,1n,thm

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

p,opt,line,pwls

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

p,opt,bb,pre

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

p,opt,pcg,hz

Problem 11.9 *The formula for γ_n^{HZ} given in [66, p. 2] was for the case $\boldsymbol{P} = \boldsymbol{I}$. Use the coordinate transformation ideas of §11.3.8 to derive the preconditioned version (11.8.13).*

11.13 Bibliography

bertsekas:82
[1] D. P. Bertsekas. *Constrained optimization and Lagrange multiplier methods*. New York: Academic-Press, 1982. URL: http://www.athenasc.com/ (cit. on pp. 11.2, 11.11, 11.19).

polyak:87
[2] B. T. Polyak. *Introduction to optimization*. New York: Optimization Software Inc, 1987 (cit. on pp. 11.2, 11.5, 11.7, 11.10, 11.18).

bertsekas:99
[3] D. P. Bertsekas. *Nonlinear programming*. 2nd ed. Belmont: Athena Scientific, 1999. URL: http://www.athenasc.com/nonlinbook.html (cit. on pp. 11.2, 11.9, 11.21).

boyd:04
[4] S. Boyd and L. Vandenberghe. *Convex optimization*. UK: Cambridge, 2004. URL: http://web.stanford.edu/~boyd/cvxbook.html (cit. on p. 11.2).

nesterov:04
[5] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*. Springer, 2004. DOI: 10.1007/978-1-4419-8853-9 (cit. on pp. 11.2, 11.15, 11.17).

lange:13
[6] K. Lange. *Optimization*. Springer, 2013. DOI: 10.1007/978-1-4614-5838-8 (cit. on p. 11.2).

migdalas:03:noa
[7] A. Migdalas, G. Toraldo, and V. Kumar. “Nonlinear optimization and parallel computing.” In: *Parallel computing* 29.4 (Apr. 2003), 375–91. DOI: 10.1016/S0167-8191(03)00013-9 (cit. on p. 11.2).

lange:14:abs
[8] K. Lange, E. C. Chi, and H. Zhou. “A brief survey of modern optimization for statisticians.” In: *Int. Stat. Review* 82.1 (Apr. 2014), 46–70. DOI: 10.1111/insr.12022 (cit. on p. 11.2).

chambolle:16:ait
[9] A. Chambolle and T. Pock. “An introduction to continuous optimization for imaging.” In: *Acta Numerica* 25 (2016), 161–319. DOI: 10.1017/S096249291600009X (cit. on p. 11.2).

luenberger:69
[10] D. G. Luenberger. *Optimization by vector space methods*. New York: Wiley, 1969. URL: http://books.google.com/books?id=lZU0CAH4RccC (cit. on pp. 11.2, 11.4).

jacobson:07:aet
[11] M. W. Jacobson and J. A. Fessler. “An expanded theoretical treatment of iteration-dependent majorize-minimize algorithms.” In: *IEEE Trans. Im. Proc.* 16.10 (Oct. 2007), 2411–22. DOI: 10.1109/TIP.2007.904387 (cit. on p. 11.3).

jacobson:04:pom
[12] M. W. Jacobson and J. A. Fessler. *Properties of MM algorithms on convex feasible sets: extended version*. Tech. rep. 353. Univ. of Michigan, Ann Arbor, MI, 48109-2122: Comm. and Sign. Proc. Lab., Dept. of EECS, Nov. 2004. URL: http://web.eecs.umich.edu/~fessler/papers/files/tr/04,jacobson.pdf (cit. on p. 11.3).

meyer:76:scf
[13] R. R. Meyer. “Sufficient conditions for the convergence of monotonic mathematical programming algorithms.” In: *J. Comput. System. Sci.* 12.1 (1976), 108–21. DOI: '10.1016/S0022-0000(76)80021-9' (cit. on p. 11.4).

nelder:65:asm
[14] J. A. Nelder and R. Mead. “A simplex method for function minimization.” In: *Computer Journal* 7.4 (1965), 308–13. DOI: 10.1093/comjnl/7.4.308 (cit. on p. 11.4).

kolda:03:obd
[15] T. G. Kolda, R. M. Lewis, and V. Torczon. “Optimization by direct search: new perspectives on some classical and modern methods.” In: *SIAM Review* 45.3 (2003), 385–482. DOI: 10.1137/S003614450242889 (cit. on p. 11.4).

richardson:72:bbi
[16] W. H. Richardson. “Bayesian-based iterative method of image restoration.” In: *J. Opt. Soc. Am.* 62.1 (Jan. 1972), 55–9. DOI: 10.1364/JOSA.62.000055 (cit. on p. 11.4).

lucy:74:ait
[17] L. Lucy. “An iterative technique for the rectification of observed distributions.” In: *The Astronomical Journal* 79.6 (June 1974), 745–54. URL: http://adsabs.harvard.edu/cgi-bin/nph-bib_query?bibcode=1974AJ.....79..745L (cit. on p. 11.4).

ortega:70
[18] J. M. Ortega and W. C. Rheinboldt. *Iterative solution of nonlinear equations in several variables*. New York: Academic, 1970. DOI: 10.1137/1.9780898719468 (cit. on p. 11.6).

levitin:66:cmm
[19] E. S. Levitin and B. T. Polyak. “Constrained minimization methods.” In: *USSR Computational Mathematics and Mathematical Physics* 6.5 (1966), 1–50. DOI: '10.1016/0041-5553(66)90114-5' (cit. on pp. 11.7, 11.8, 11.10).

beck:09:fgb
[20] A. Beck and M. Teboulle. “Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems.” In: *IEEE Trans. Im. Proc.* 18.11 (Nov. 2009), 2419–34. DOI: 10.1109/TIP.2009.2028250 (cit. on p. 11.8).

drori:14:pof
[21] Y. Drori and M. Teboulle. "Performance of first-order methods for smooth convex minimization: A novel approach." In: *Mathematical Programming* 145.1-2 (June 2014), 451–82. DOI: 10.1007/s10107-013-0653-0 (cit. on pp. 11.8, 11.15).

lange:99
[22] K. Lange. *Numerical analysis for statisticians*. New York: Springer-Verlag, 1999 (cit. on pp. 11.9, 11.18, 11.19).

albaali:96:oto
[23] M. Al-Baali and R. Fletcher. "On the order of convergence of preconditioned nonlinear conjugate gradient methods." In: *SIAM J. Sci. Comp.* 17.3 (May 1996), 658–65. DOI: 10.1137/S1064827591194303 (cit. on pp. 11.9, 11.17).

jarre:16:sef
[24] F. Jarre and P. L. Toint. "Simple examples for the failure of Newton's method with line search for strictly convex minimization." In: *Mathematical Programming* 158.1 (July 2016), 23–34. DOI: 10.1007/s10107-015-0913-2 (cit. on p. 11.10).

cartis:19:urm
[25] C. Cartis, N. I. Gould, and P. L. Toint. "Universal regularization methods: varying the power, the smoothness and the accuracy." In: *SIAM J. Optim.* 29.1 (Jan. 2019), 595–615. DOI: 10.1137/16m1106316 (cit. on p. 11.10).

cauchy:1847:mgp
[26] A. Cauchy. "Methode générale pour la résolution des systems d'équations simultanées." In: *Comp. Rend. Sci. Paris* 25 (1847), 536–8 (cit. on p. 11.10).

greenbaum:97
[27] A. Greenbaum. *Iterative methods for solving linear systems*. Philadelphia: Soc. Indust. Appl. Math., 1997 (cit. on pp. 11.11, 11.16, 11.17).

gonzaga:16:otw
[28] C. C. Gonzaga. "On the worst case performance of the steepest descent algorithm for quadratic functions." In: *Mathematical Programming* 160.1 (Nov. 2016), 307–20. DOI: 10.1007/s10107-016-0984-8 (cit. on p. 11.12).

deklerk:17:otw
[29] E. de Klerk, Francois Glineur, and A. B. Taylor. "On the worst-case complexity of the gradient method with exact line search for smooth strongly convex functions." In: *Optics Letters* 11.7 (Oct. 2017), 1185–99. DOI: 10.1007/s11590-016-1087-4 (cit. on p. 11.12).

kiefer:53:sms
[30] J. Kiefer. "Sequential minimax search for a maximum." In: *Proc. Amer. Math. Soc.* 4.3 (1953), 502–6. DOI: 10.2307/2032161 (cit. on p. 11.12).

press:88
[31] W. H. Press et al. *Numerical recipes in C*. New York: Cambridge Univ. Press, 1988 (cit. on pp. 11.12, 11.16, 11.20).

hager:05:anc
[32] W. W. Hager and H. Zhang. "A new conjugate gradient method with guaranteed descent and an efficient line search." In: *SIAM J. Optim.* 16.1 (2005), 170–92. DOI: 10.1137/030601880 (cit. on pp. 11.12, 11.17).

wolfe:69:ccf
[33] P. Wolfe. "Convergence conditions for ascent methods." In: *SIAM Review* 11.2 (1969), 226–35. DOI: 10.1137/1011036 (cit. on p. 11.12).

wolfe:71:ccf
[34] P. Wolfe. "Convergence conditions for ascent methods. II: Some corrections." In: *SIAM Review* 13.2 (1971), 185–8. DOI: 10.1137/1013035 (cit. on p. 11.12).

armijo:66:mof
[35] L. Armijo. "Minimization of functions having continuous derivatives." In: *Pacific J. Math* 16.1 (1966), 1–3. URL: http://projecteuclid.org/euclid.pjm/1102995080 (cit. on p. 11.12).

odonoghue:15:arf
[36] B. O'Donoghue and E. Candes. "Adaptive restart for accelerated gradient schemes." In: *Found. Comp. Math.* 15.3 (June 2015), 715–32. DOI: 10.1007/s10208-013-9150-3 (cit. on p. 11.13).

barzilai:88:tps
[37] J. Barzilai and J. Borwein. "Two-point step size gradient methods." In: *IMA J. Numerical Analysis* 8.1 (1988), 141–8. DOI: 10.1093/imanum/8.1.141 (cit. on p. 11.13).

raydan:93:otb
[38] M. Raydan. "On the Barzilai and Borwein choice of steplength for the gradient method." In: *IMA J. Numerical Analysis* 13.3 (1993), 321–6. DOI: 10.1093/imanum/13.3.321 (cit. on pp. 11.13, 11.14).

raydan:97:tba
[39] M. Raydan. "The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem." In: *SIAM J. Optim.* 7.1 (1997), 26–33. DOI: 10.1137/S1052623494266365 (cit. on pp. 11.13, 11.14).

fletcher:05:otb
[40] R. Fletcher. "On the Barzilai-Borwein method, optimization and control with applications." In: *Appl. Optics* 96.II (2005), 235–56. DOI: 10.1007/0-387-24255-4_10 (cit. on p. 11.13).

dai:02:rlc
[41] Y-H. Dai and L-Z. Liao. "R-linear convergence of the Barzilai and Borwein gradient method." In: *IMA J. Numerical Analysis* 22.1 (2002), 1–10. DOI: 10.1093/imanum/22.1.1 (cit. on p. 11.14).

grippo:86:anl [42] L. Grippo, F. Lampariello, and S. Lucidi. "A nonmonotone line search technique for Newton's method." In: *SIAM J. Numer. Anal.* 23.4 (1986), 707–16. DOI: 10.1137/0723046 (cit. on p. 11.14).

grippo:91:aco [43] L. Grippo, F. Lampariello, and S. Lucidi. "A class of nonmonotone stabilization methods in unconstrained optimization." In: *Numerische Mathematik* 59.5 (1991), 779–805. DOI: 10.1007/BF01385810 (cit. on p. 11.14).

yuan:06:ssf [44] Y. Yuan. *Step-sizes for the gradient method*. 2006. URL: ftp://lsec.cc.ac.cn/pub/yyx/papers/p0504.pdf (cit. on p. 11.14).

nesterov:83:amf [45] Y. Nesterov. "A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$." In: *Dokl. Akad. Nauk. USSR* 269.3 (1983), 543–7 (cit. on pp. 11.14, 11.15).

nesterov:05:smo [46] Y. Nesterov. "Smooth minimization of non-smooth functions." In: *Mathematical Programming* 103.1 (May 2005), 127–52. DOI: 10.1007/s10107-004-0552-5 (cit. on pp. 11.14, 11.15).

zuo:11:aga [47] W. Zuo and Z. Lin. "A generalized accelerated proximal gradient approach for total-variation-based image restoration." In: *IEEE Trans. Im. Proc.* 20.10 (Oct. 2011), 2748–59. DOI: 10.1109/TIP.2011.2131665 (cit. on p. 11.14).

kim:13:axr [48] D. Kim, S. Ramani, and J. A. Fessler. "Accelerating X-ray CT ordered subsets image reconstruction with Nesterov's first-order methods." In: *Proc. Intl. Mtg. on Fully 3D Image Recon. in Rad. and Nuc. Med.* 2013, 22–5. URL: http://web.eecs.umich.edu/~fessler/papers/files/proc/13/web/kim-13-axr.pdf (cit. on p. 11.14).

kim:15:cos [49] D. Kim, S. Ramani, and J. A. Fessler. "Combining ordered subsets and momentum for accelerated X-ray CT image reconstruction." In: *IEEE Trans. Med. Imag.* 34.1 (Jan. 2015), 167–78. DOI: 10.1109/TMI.2014.2350962 (cit. on p. 11.14).

zibetti:17:aoa [50] M. V. W. Zibetti, E. S. Helou, and D. R. Pipa. "Accelerating overrelaxed and monotone fast iterative shrinkage-thresholding algorithms with line search for sparse reconstructions." In: *IEEE Trans. Im. Proc.* 26.7 (July 2017), 3569–78. DOI: 10.1109/TIP.2017.2699483 (cit. on p. 11.15).

baes:14:aap [51] M. Baes and M. Bürgisser. "An acceleration procedure for optimal first-order methods." In: *Optim. Meth. Software* 29.3 (2014), 610–28. DOI: 10.1080/10556788.2013.835812 (cit. on p. 11.15).

kim:16:ofo [52] D. Kim and J. A. Fessler. "Optimized first-order methods for smooth convex minimization." In: *Mathematical Programming* 159.1 (Sept. 2016), 81–107. DOI: 10.1007/s10107-015-0949-3 (cit. on p. 11.15).

drori:17:tei [53] Y. Drori. "The exact information-based complexity of smooth convex minimization." In: *J. Complexity* 39 (Apr. 2017), 1–16. DOI: 10.1016/j.jco.2016.11.001 (cit. on p. 11.15).

rumelhart:86:lrb [54] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning representations by back-propagating errors." In: *Nature* 323.6088 (Oct. 1986), 533–6. DOI: 10.1038/323533a0 (cit. on p. 11.15).

sutskever:13:oti [55] I. Sutskever et al. "On the importance of initialization and momentum in deep learning." In: *Proc. Intl. Conf. Mach. Learn.* 2013, 1139–47. URL: http://www.cs.utoronto.ca/~ilya/pubs/2013/1051_2.pdf (cit. on p. 11.15).

daniel:67:tcg [56] J. W. Daniel. "The conjugate gradient method for linear and nonlinear operator equations." In: *SIAM J. Numer. Anal.* 4.1 (1967), 10–26. DOI: 10.1137/0704002 (cit. on p. 11.16).

golub:89:sho [57] G. H. Golub and D. P. O'Leary. "Some history of the conjugate gradient and Lanczos methods." In: *SIAM Review* 31.1 (Mar. 1989), 50–102. DOI: 10.1137/1031003 (cit. on p. 11.16).

hestenes:52:moc [58] M. Hestenes and E. Stiefel. "Methods of conjugate gradients for solving linear systems." In: *J. Research Nat. Bur. Standards* 49.6 (Dec. 1952), 409–36. URL: http://archive.org/details/jresv49n6p409 (cit. on p. 11.16).

polak:69:nsl [59] E. Polak and G. Ribière. "Note sur la convergence de méthodes de directions conjuguées." In: *Rev. Française Informat. Recherche Opérationnelle* 3 (1969), 35–43 (cit. on p. 11.16).

polyak:69:tcg [60] B. T. Polyak. "The conjugate gradient method in extremal problems." In: *USSR Comp. Math. Math. Phys.* 9 (1969), 94–112 (cit. on p. 11.16).

mumcuoglu:94:fgb [61] E. U. Mumcuoglu et al. "Fast gradient-based methods for Bayesian reconstruction of transmission and emission PET images." In: *IEEE Trans. Med. Imag.* 13.3 (Dec. 1994), 687–701. DOI: 10.1109/42.363099 (cit. on p. 11.16).

fessler:99:cgp
[62] J. A. Fessler and S. D. Booth. "Conjugate-gradient preconditioning methods for shift-variant PET image reconstruction." In: *IEEE Trans. Im. Proc.* 8.5 (May 1999), 688–99. DOI: 10.1109/83.760336 (cit. on pp. 11.16, 11.17).

dai:99:anc
[63] Y. H. Dai and Y. Yuan. "A nonlinear conjugate gradient method with a strong global convergence property." In: *SIAM J. Optim.* 10.1 (1999), 177–82. DOI: 10.1137/S1052623497318992 (cit. on p. 11.17).

fletcher:64:fmb
[64] R. Fletcher and C. M. Reeves. "Function minimization by conjugate gradients." In: *Comput. J* 7.2 (1964), 149–54. DOI: 10.1093/comjnl/7.2.149 (cit. on p. 11.17).

hager:06:a8c
[65] W. W. Hager and H. Zhang. "Algorithm 851: CG-DESCENT, a conjugate gradient method with guaranteed descent." In: *ACM Trans. Math. Software* 32.1 (Mar. 2006), 113–37. DOI: 10.1145/1132973.1132979 (cit. on p. 11.17).

hager:06:aso
[66] W. W. Hager and H. Zhang. "A survey of nonlinear conjugate gradient methods." In: *Pacific J Optimization* 2.1 (Jan. 2006), 35–58. URL: http://www.ybook.co.jp/online2/pjov6-1.html (cit. on pp. 11.17, 11.23).

broyden:96:ant
[67] C. G. Broyden. "A new taxonomy of conjugate gradient methods." In: *Computers & Mathematics with Applications* 31.4-5 (1996), 7–17. DOI: 10.1016/0898-1221(95)00211-1 (cit. on p. 11.17).

dai:99:cpo
[68] Y. Dai et al. "Convergence properties of nonlinear conjugate gradient methods." In: *SIAM J. Optim.* 10.2 (1999), 345–58. DOI: 10.1137/S1052623494268443 (cit. on p. 11.17).

dai:01:ncc
[69] Y. H. Dai and L. Z. Liao. "New conjugacy conditions and related nonlinear conjugate gradient methods." In: *Appl. Math. Optim.* 43.1 (Jan. 2001), 87–101. DOI: 10.1007/s002450010019 (cit. on p. 11.17).

labat:07:coc
[70] C. Labat and Jerome Idier. "Convergence of conjugate gradient methods with a closed-form stepsize formula." In: *J. Optim. Theory Appl.* 136.1 (Jan. 2007), 43–60. DOI: 10.1007/s10957-007-9306-x (cit. on p. 11.17).

labat:08:coc
[71] C. Labat and Jerome Idier. "Convergence of conjugate gradient methods with a closed-form stepsize formula." In: *J. Optim. Theory Appl.* 136 (2008), 43–60. DOI: 10.1007/s10957-007-9306-x (cit. on p. 11.17).

andrei:09:hcg
[72] N. Andrei. "Hybrid conjugate gradient algorithm for unconstrained optimization." In: *J. Optim. Theory Appl.* 141.2 (May 2009), 249–64. DOI: 10.1007/s10957-008-9505-0 (cit. on p. 11.17).

dai:10:ncg
[73] Y-H. Dai. *Nonlinear conjugate gradient methods*. Wiley Encyclopedia of Operations Research and Management Science, edited by James J. Cochran. 2010. DOI: 10.1002/9780470400531.eorms0183 (cit. on p. 11.17).

narushima:11:att
[74] Y. Narushima, H. Yabe, and J. A. Ford. "A three-term conjugate gradient method with sufficient descent property for unconstrained optimization." In: *SIAM J. Optim.* 21.1 (2011), 212–30. DOI: 10.1137/080743573 (cit. on p. 11.17).

andrei:13:acg
[75] N. Andrei. "Another conjugate gradient algorithm with guaranteed descent and conjugacy conditions for large-scale unconstrained optimization." In: *J. Optim. Theory Appl.* 159.1 (Oct. 2013), 159–82. DOI: 10.1007/s10957-013-0285-9 (cit. on p. 11.17).

dai:13:anc
[76] Y. Dai and C. Kou. "A nonlinear conjugate gradient algorithm with an optimal property and an improved Wolfe line search." In: *SIAM J. Optim.* 23.1 (2013), 296–320. DOI: 10.1137/100813026 (cit. on p. 11.17).

dai:17:coh
[77] Z. Dai. "Comments on hybrid conjugate gradient algorithm for unconstrained optimization." In: *J. Optim. Theory Appl.* 175.1 (Oct. 2017), 286–91. DOI: 10.1007/s10957-017-1172-6 (cit. on p. 11.17).

powell:86:cpo
[78] M. J. D. Powell. "Convergence properties of algorithms for nonlinear optimization." In: *SIAM Review* 28.4 (Dec. 1986), 487–500. DOI: 10.1137/1028154 (cit. on p. 11.17).

joly:93:ccg
[79] P. Joly and Gerard Meurant. "Complex conjugate gradient methods." In: *Numer. Algorithms* 4.3 (1993), 379–406. DOI: 10.1007/BF02145754 (cit. on p. 11.17).

freund:92:cgt
[80] R. W. Freund. "Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices." In: *SIAM J. Sci. Stat. Comp.* 13.1 (1992), 425–48. DOI: 10.1137/0913023 (cit. on p. 11.17).

bunsegerstner:99:oac
[81] A. Bunse-Gerstner and R. Stover. "On a conjugate gradient-type method for solving complex symmetric linear systems." In: *Linear Algebra and its Applications* 287.1-3 (Jan. 1999), 105–23. DOI: 10.1016/S0024-3795(98)10091-5 (cit. on p. 11.17).

chouzenoux:11:amm
[82] E. Chouzenoux, J. Idier, and Said Moussaoui. “A majorize-minimize strategy for subspace optimization applied to image restoration.” In: *IEEE Trans. Im. Proc.* 20.6 (June 2011), 1517–28. DOI: 10.1109/TIP.2010.2103083 (cit. on p. 11.17).

gu:04:msd-1
[83] T. Gu et al. “Multiple search direction conjugate gradient method I: methods and their propositions.” In: *Int. J. of Computer Mathematics* 81.9 (Sept. 2004), 1133–43. DOI: 10.1080/00207160410001712305 (cit. on p. 11.17).

gu:04:msd-2
[84] T. Gu et al. “Multiple search direction conjugate gradient method II: theory and numerical experiments.” In: *Int. J. of Computer Mathematics* 81.10 (Oct. 2004), 1289–307. DOI: 10.1080/00207160412331289065 (cit. on p. 11.17).

bridson:06:amp
[85] R. Bridson and C. Greif. “A multi-preconditioned conjugate gradient algorithm.” In: *SIAM J. Matrix. Anal. Appl.* 27.4 (2006), 1056–68. DOI: 10.1137/040620047 (cit. on p. 11.17).

florescu:14:amm
[86] A. Florescu et al. “A majorize-minimize memory gradient method for complex-valued inverse problems.” In: *Signal Processing* 103 (Oct. 2014), 285–95. DOI: 10.1016/j.sigpro.2013.09.026 (cit. on p. 11.17).

elad:07:cas
[87] M. Elad, B. Matalon, and M. Zibulevsky. “Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization.” In: *Applied and Computational Harmonic Analysis* 23.3 (Nov. 2007), 346–67. DOI: 10.1016/j.acha.2007.02.002 (cit. on p. 11.17).

davidon:59:vmm
[88] W. C. Davidon. *Variable metric methods for minimization*. Tech. rep. ANL-5990. Argonne National Laboratory, USA: AEC Research and Development Report, 1959 (cit. on p. 11.18).

khalfan:93:ata
[89] H. F. Khalfan, R. H. Byrd, and R. B. Schnabel. “A theoretical and experimental study of the symmetric rank-one update.” In: *SIAM J. Optim.* 3.1 (Feb. 1993), 1–24. DOI: 10.1137/0803001 (cit. on p. 11.18).

more:94:lsa
[90] J. J. Moré and D. J. Thuente. “Line search algorithms with guaranteed sufficient decrease.” In: *ACM Trans. Math. Software* 20.3 (Sept. 1994), 286–307. DOI: 10.1145/192115.192132 (cit. on p. 11.18).

kolda:98:bwu
[91] T. G. Kolda, D. P. O’Leary, and L. Nazareth. “BFGS with update skipping and varying memory.” In: *SIAM J. Optim.* 8.4 (1998), 1060–83. DOI: 10.1137/S1052623496306450 (cit. on p. 11.18).

dennis:74:aco
[92] J. E. Dennis and J. Moré. “A characterization of superlinear convergence and its application to quasi-newton methods.” In: *Mathematics of Computation* 28.126 (Apr. 1974), 549–60. URL: http://www.jstor.org/stable/2005926 (cit. on p. 11.18).

zhu:97:a7l
[93] C. Zhu et al. “Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization.” In: *ACM Trans. Math. Software* 23.4 (Dec. 1997), 550–60. DOI: 10.1145/279232.279236 (cit. on p. 11.19).

lange:95:aqn
[94] K. Lange. “A Quasi-Newton acceleration of the EM Algorithm.” In: *Statistica Sinica* 5.1 (Jan. 1995), 1–18. URL: http://www3.stat.sinica.edu.tw/statistica/j5n1/j5n11/j5n11.htm (cit. on p. 11.19).

jamshidian:97:aot
[95] M. Jamshidian and R. I. Jennrich. “Acceleration of the EM algorithm by using quasi-Newton methods.” In: *J. Royal Stat. Soc. Ser. B* 59.3 (1997), 569–87. DOI: 10.1111/1467-9868.00083 (cit. on p. 11.19).

klose:03:qnm
[96] A. D. Klose and A. H. Hielscher. “Quasi-Newton methods in optical tomographic image reconstruction.” In: *Inverse Prob.* 19.2 (Apr. 2003), 387–409. DOI: 10.1088/0266-5611/19/2/309 (cit. on p. 11.19).

ramirezgiraldo:11:npi
[97] J. C. Ramirez-Giraldo et al. “Nonconvex prior image constrained compressed sensing (NCPICCS): Theory and simulations on perfusion CT.” In: *Med. Phys.* 38.4 (Apr. 2011), 2157–67. DOI: 10.1118/1.3560878 (cit. on p. 11.19).

tsai:18:fqn
[98] Y-J. Tsai et al. “Fast quasi-Newton algorithms for penalized reconstruction in emission tomography and further improvements via preconditioning.” In: *IEEE Trans. Med. Imag.* 37.4 (Apr. 2018), 1000–10. DOI: 10.1109/tmi.2017.2786865 (cit. on p. 11.19).

ziskind:88:mll
[99] I. Ziskind and M. Wax. “Maximum likelihood localization of multiple sources by alternating projection.” In: *IEEE Trans. Acoust. Sp. Sig. Proc.* 36.10 (Oct. 1988), 1553–60 (cit. on p. 11.21).

hathaway:91:gcm
[100] R. J. Hathaway and J. C. Bezdek. “Grouped coordinate minimization using Newton’s method for inexact minimization in one vector coordinate.” In: *J. Optim. Theory Appl.* 71.3 (Dec. 1991), 503–16. DOI: 10.1007/BF00941400 (cit. on p. 11.21).

jensen:91:gca
[101] S. T. Jensen, S. Johansen, and S. L. Lauritzen. “Globally convergent algorithms for maximizing a likelihood function.” In: *Biometrika* 78.4 (Dec. 1991), 867–77. DOI: 10.1093/biomet/78.4.867 (cit. on p. 11.21).

clinthorne:94:acd
[102] N. H. Clinthorne. "A constrained dual-energy reconstruction method for material-selective transmission tomography." In: *Nucl. Instr. Meth. Phys. Res. A.* 353.1 (Dec. 1994), 347–8. DOI: 10.1016/0168-9002(94)91673-X (cit. on p. 11.21).

grippo:99:gcb
[103] L. Grippo and M. Sciandrone. "Globally convergent block-coordinate techniques for unconstrained optimization." In: *Optim. Meth. Software* 10.4 (Apr. 1999), 587–637. DOI: 10.1080/10556789908805730 (cit. on p. 11.21).

sauer:95:pco
[104] K. D. Sauer, S. Borman, and C. A. Bouman. "Parallel computation of sequential pixel updates in statistical tomographic reconstruction." In: *Proc. IEEE Intl. Conf. on Image Processing*. Vol. 3. 1995, 93–6. DOI: 10.1109/ICIP.1995.537422 (cit. on p. 11.21).

fessler:97:gca
[105] J. A. Fessler et al. "Grouped-coordinate ascent algorithms for penalized-likelihood transmission image reconstruction." In: *IEEE Trans. Med. Imag.* 16.2 (Apr. 1997), 166–75. DOI: 10.1109/42.563662 (cit. on p. 11.21).

fessler:97:gcd
[106] J. A. Fessler. "Grouped coordinate descent algorithms for robust edge-preserving image restoration." In: *Proc. SPIE 3170 Im. Recon. and Restor. II*. 1997, 184–94. DOI: 10.1117/12.279713 (cit. on p. 11.21).

fessler:11:abc
[107] J. A. Fessler and D. Kim. "Axial block coordinate descent (ABCD) algorithm for X-ray CT image reconstruction." In: *Proc. Intl. Mtg. on Fully 3D Image Recon. in Rad. and Nuc. Med*. 2011, 262–5. URL: http://www.fully3d.org (cit. on p. 11.21).

mcgaffin:14:fep
[108] M. G. McGaffin and J. A. Fessler. "Fast edge-preserving image denoising via group coordinate descent on the GPU." In: *Proc. SPIE 9020 Computational Imaging XII*. 2014, 90200P. DOI: 10.1117/12.2042593 (cit. on p. 11.21).

tseng:01:coa
[109] P. Tseng. "Convergence of a block coordinate descent methods for nondifferentiable minimization." In: *J. Optim. Theory Appl.* 109 (2001), 475–94. DOI: 10.1023/A:1017501703105 (cit. on p. 11.21).

nesterov:12:eoc
[110] Y. Nesterov. "Efficiency of coordinate descent methods on huge-scale optimization problems." In: *SIAM J. Optim.* 22.2 (2012), 341–62. DOI: 10.1137/100802001 (cit. on p. 11.21).

xu:13:abc
[111] Y. Xu and W. Yin. "A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion." In: *SIAM J. Imaging Sci.* 6.3 (2013), 1758–89. DOI: 10.1137/120887795 (cit. on p. 11.21).

shi:16:abc
[112] Z. Shi and R. Liu. *A better convergence analysis of the block coordinate descent method for large scale machine learning*. 2016. URL: http://arxiv.org/abs/1608.04826 (cit. on p. 11.21).

razaviyayn:13:auc
[113] M. Razaviyayn, M. Hong, and Z. Luo. "A unified convergence analysis of block successive minimization methods for nonsmooth optimization." In: *SIAM J. Optim.* 23.2 (2013), 1126–53. DOI: 10.1137/120891009 (cit. on p. 11.21).

bolte:14:pal
[114] J. Bolte, S. Sabach, and M. Teboulle. "Proximal alternating linearized minimization for nonconvex and nonsmooth problems." In: *Mathematical Programming* 146.1 (Aug. 2014), 459–94. DOI: 10.1007/s10107-013-0701-9 (cit. on p. 11.21).

yun:14:oti
[115] S. Yun. "On the iteration complexity of cyclic coordinate gradient descent methods." In: *SIAM J. Optim.* 24.3 (2014), 1567–80. DOI: 10.1137/130937755 (cit. on pp. 11.21, 11.22).

chouzenoux:16:abc
[116] E. Chouzenoux, J-C. Pesquet, and A. Repetti. "A block coordinate variable metric forward-backward algorithm." In: *J. of Global Optimization* 66.3 (Nov. 2016), 457–85. DOI: 10.1007/s10898-016-0405-9 (cit. on p. 11.21).

nocedal:99
[117] J. Nocedal and S. J. Wright. *Numerical optimization*. New York: Springer, 1999 (cit. on p. 11.21).

gullberg:87:mer
[118] G. Gullberg and B. M. W. Tsui. "Maximum entropy reconstruction with constraints: iterative algorithms for solving the primal and dual programs." In: *Proc. Tenth Intl. Conf. on Information Processing in Medical Im.* Ed. by C N de Graaf and M A Viergever. New York: Plenum Press, 1987, pp. 181–200 (cit. on p. 11.21).

press:92
[119] W. H. Press et al. *Numerical recipes in C*. 2nd ed. New York: Cambridge Univ. Press, 1992 (cit. on pp. 11.21, 11.23).

lu:98:crw
[120] H. H-S. Lu, C-M. Chen, and I-H. Yang. "Cross-reference weighted least square estimates for positron emission tomography." In: *IEEE Trans. Med. Imag.* 17.1 (Feb. 1998), 1–8. DOI: 10.1109/42.668690 (cit. on p. 11.21).

sauer:93:alu
[121] K. Sauer and C. Bouman. "A local update strategy for iterative reconstruction from projections." In: *IEEE Trans. Sig. Proc.* 41.2 (Feb. 1993), 534–48. DOI: 10.1109/78.193196 (cit. on p. 11.21).

qi:98:hr3
[122] J. Qi et al. “High resolution 3D Bayesian image reconstruction using the microPET small-animal scanner.” In: *Phys. Med. Biol.* 43.4 (Apr. 1998), 1001–14. DOI: 10.1088/0031-9155/43/4/027 (cit. on p. 11.21).

bouman:93:fnm
[123] C. Bouman and K. Sauer. “Fast numerical methods for emission and transmission tomographic reconstruction.” In: *Proc. 27th Conf. Info. Sci. Sys., Johns Hopkins*. 1993, 611–6 (cit. on p. 11.22).

bouman:95:tma
[124] C. A. Bouman, K. Sauer, and S. S. Saquib. “Tractable models and efficient algorithms for Bayesian tomography.” In: *Proc. IEEE Conf. Acoust. Speech Sig. Proc.* Vol. 5. 1995, 2907–10. DOI: 10.1109/ICASSP.1995.479453 (cit. on p. 11.22).

fessler:94:sag
[125] J. A. Fessler and A. O. Hero. “Space-alternating generalized expectation-maximization algorithm.” In: *IEEE Trans. Sig. Proc.* 42.10 (Oct. 1994), 2664–77. DOI: 10.1109/78.324732 (cit. on p. 11.22).

luo:92:otc
[126] Z. Q. Luo and P. Tseng. “On the convergence of the coordinate descent method for convex differentiable minimization.” In: *J. Optim. Theory Appl.* 72.1 (Jan. 1992), 7–35. DOI: 10.1007/BF00939948 (cit. on p. 11.22).

powell:73:osd
[127] M. J. D. Powell. “On search directions for minimization algorithms.” In: *Mathematical Programming* 4.1 (1973), 193–201. DOI: 10.1007/BF01584660 (cit. on p. 11.22).

luo:93:otc
[128] Z-Q. Luo and P. Tseng. “On the convergence rate of dual ascent methods for linearly constrained convex minimization.” In: *Math. Oper. Res.* 18.4 (Nov. 1993), 846–67. URL: http://www.jstor.org/stable/3690126 (cit. on p. 11.22).

kirkpatrick:83:obs
[129] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. “Optimization by simulated annealing.” In: *Science* 220.4598 (May 1983), 671–80 (cit. on p. 11.23).

fessler:92:sac
[130] J. A. Fessler. “Segmented attenuation correction for PET using ICM.” In: *Proc. IEEE Nuc. Sci. Symp. Med. Im. Conf.* Vol. 2. 1992, 1182–4. DOI: 10.1109/NSSMIC.1992.301040 (cit. on p. 11.23).

kudo:00:ana
[131] H. Kudo and H. Nakamura. “A new approach to SPECT attenuation correction without transmission measurements.” In: *Proc. IEEE Nuc. Sci. Symp. Med. Im. Conf.* Vol. 2. 2000, 13/58–62. DOI: 10.1109/NSSMIC.2000.949991 (cit. on p. 11.23).

kudo:00:sam
[132] H. Kudo and H. Nakamura. “Segmented attenuation map reconstruction from incomplete transmission data.” In: *Proc. IEEE Nuc. Sci. Symp. Med. Im. Conf.* Vol. 2. 2000, 13/1–5. DOI: 10.1109/NSSMIC.2000.949979 (cit. on p. 11.23).

xu:91:asa
[133] E. Z. Xu et al. “A segmented attenuation correction for PET.” In: *J. Nuc. Med.* 32.1 (Jan. 1991), 161–5. URL: http://jnm.snmjournals.org/cgi/content/abstract/32/1/161 (cit. on p. 11.23).

bettinardi:99:aac
[134] V. Bettinardi et al. “An automatic classification technique for attenuation correction in positron emission tomography.” In: *Eur. J. Nuc. Med.* 26.5 (May 1999), 447–58. DOI: 10.1007/s002590050410 (cit. on p. 11.23).

webb:89:srb
[135] S. Webb. “SPECT reconstruction by simulated annealing.” In: *Phys. Med. Biol.* 34.3 (Mar. 1989), 259–82. DOI: 10.1088/0031-9155/34/3/001 (cit. on p. 11.23).

kearfott:90:sai
[136] K. J. Kearfott and S. E. Hill. “Simulated annealing image reconstruction method for a pinhole aperture single photon emission computed tomograph (SPECT).” In: *IEEE Trans. Med. Imag.* 9.2 (June 1990), 128–43. DOI: 10.1109/42.56337 (cit. on p. 11.23).

sundermann:94:sai
[137] E. Sundermann, I. Lemahieu, and P. Desmedt. “Simulated annealing image reconstruction for PET.” In: *Intl. Conf. Med. Phys. Biomed. Eng.* Vol. 1. 1994, 175–9 (cit. on p. 11.23).

girodias:91:psa
[138] K. A. Girodias, H. H. Barrett, and R. L. Shoemaker. “Parallel simulated annealing for emission tomography.” In: *Phys. Med. Biol.* 36.7 (July 1991), 921–38. DOI: 10.1088/0031-9155/36/7/002 (cit. on p. 11.23).

metropolis:53:eos
[139] N. Metropolis et al. “Equation of state calculations by fast computing machines.” In: *J. Chem. Phys.* 21.6 (June 1953), 1087–92. DOI: 10.1063/1.1699114 (cit. on p. 11.23).

blake:89:cot
[140] A. Blake. “Comparison of the efficiency of deterministic and stochastic algorithms for visual reconstruction.” In: *IEEE Trans. Patt. Anal. Mach. Int.* 11.1 (Jan. 1989), 2–12. DOI: 10.1109/34.23109 (cit. on p. 11.23).

geiger:91:pad
[141] D. Geiger and F. Girosi. “Parallel and deterministic algorithms from MRF’s: Surface reconstruction.” In: *IEEE Trans. Patt. Anal. Mach. Int.* 13.5 (May 1991), 401–12. DOI: 10.1109/34.134040 (cit. on p. 11.23).

bilbro:92:mfa
[142] G. L. Bilbro et al. “Mean field annealing: A formalism for constructing GNC-like algorithms.” In: *IEEE Trans. Neural Net.* 3.1 (Jan. 1992), 131–8. DOI: 10.1109/72.105426 (cit. on p. 11.23).

bedini:93:ama
[143] L. Bedini et al. “A mixed-annealing algorithm for edge preserving image reconstruction using a limited number of projections.” In: *Signal Processing* 32.3 (June 1993), 397–408. DOI: 10.1016/0165-1684(93)90009-Y (cit. on p. 11.23).

gindi:93:brf
[144] G. Gindi et al. “Bayesian reconstruction for emission tomography via deterministic annealing.” In: *Information Processing in Medical Im.* Ed. by H H Barrett and A F Gmitro. Vol. 687. Lecture Notes in Computer Science. Berlin: Springer Verlag, 1993, pp. 322–38 (cit. on p. 11.23).

berthod:95:dad
[145] M. Berthod, Z. Kato, and J. Zerubia. “DPA: a deterministic approach to the MAP problem.” In: *IEEE Trans. Im. Proc.* 4.9 (Sept. 1995), 1312–3. DOI: 10.1109/83.413175 (cit. on p. 11.23).

lee:95:bir
[146] S-J. Lee, A. Rangarajan, and G. Gindi. “Bayesian image reconstruction in SPECT using higher order mechanical models as priors.” In: *IEEE Trans. Med. Imag.* 14.4 (Dec. 1995), 669–80. DOI: 10.1109/42.476108 (cit. on p. 11.23).

snyder:95:irr
[147] W. Snyder et al. “Image relaxation: Restoration and feature extraction.” In: *IEEE Trans. Patt. Anal. Mach. Int.* 17.6 (June 1995), 620–4. DOI: 10.1109/34.387509 (cit. on p. 11.23).

nikolova:98:iol
[148] M. Nikolova, J. Idier, and A. Mohammad-Djafari. “Inversion of large-support ill-posed linear operators using a piecewise Gaussian MRF.” In: *IEEE Trans. Im. Proc.* 7.4 (Apr. 1998), 571–85. DOI: 10.1109/83.663502 (cit. on p. 11.23).

huyer:99:gob
[149] W. Huyer and A. Neumaier. “Global optimization by multilevel coordinate search.” In: *J. Global Optimization* 14.4 (June 1999), 331–55. URL: http://www.wkap.nl/issuetoc.htm/0925-5001+14+4+1999 (cit. on p. 11.23).