Transient Dynamics of Block Coordinate Descent in a Valley[*]

Martin Mohlenkamp[†] Todd Young[‡] Balázs Bárány[§]

September 25, 2018

Abstract

We investigate the transient dynamics of Block Coordinate Descent algorithms in valleys of the optimization landscape. Iterates converge linearly to a vicinity of the valley floor and then progress in a zig-zag fashion along the direction of the valley floor. When the valley sides are symmetric, the rate of convergence to a vicinity of the valley floor appears to be no worse than 1/8, but without symmetry the rate can approach 1. Progress along the direction of the valley floor is proportional to the gradient on the valley floor and inversely proportional to the "narrowness" of the valley. We quantify narrowness using the eigenvalues of the Hessian on the valley floor and give explicit formulas for certain cases. Progress also depends on the direction of the valley with respect to the blocks of coordinates. When the valley sides are symmetric, we give an explicit formula for this dependence and use it to show that in higher dimensions nearly all directions give progress similar to the worst case direction. Finally, we observe that when starting the algorithm, the ordering of blocks in the first few steps can be important, but show that a greedy strategy with respect to objective function improvement can be a bad choice.

Keywords: Block Coordinate Descent, Alternating Least Squares, Tensor approximation, Valley, Swamp

AMS Subject Classification (2010): 37N30; 65K10

Contributions: B.B. and T.Y. began this work and wrote Sections 2.3.2, 2.3.3 and 2.3.5. M.M. and T.Y. completed the rest of the paper.

1 Introduction

Consider the generic problem of trying to find minimum points of a non-negative, differentiable objective function $f : \mathbb{R}^n \to \mathbb{R}_+$. The argument of f can be considered as a column vector $\mathbf{x}$, which can then be broken into blocks as $\mathbf{x} = (\mathbf{x}_1; \mathbf{x}_2; \dots; \mathbf{x}_d)$. Minimization with respect to a single block $\mathbf{x}_i$ while holding the other blocks fixed is often much easier than minimizing with respect to the full $\mathbf{x}$. This suggests a minimization algorithm: starting from an initial $\mathbf{x}$,

[*]This material is based upon work supported by the National Science Foundation under Grant 1418787 (M.M. and T.Y.) and grants EP/J013560/1 and OTKA K104745 (B.B.). T.Y. and B.B. thank the staff of the Warwick Mathematics Institute for their hospitality while this paper was conceived. We thank Nathaniel McClatchey for helpful discussions and reading the final version of this work.

[†]Department of Mathematics, Ohio University, Athens, OH USA.

[‡]Department of Mathematics, Ohio University, Athens, OH USA. Corresponding author `youngt@ohio.edu`

[§]Department of Stochastics, Budapest University of Technology and Economics, Budapest, Hungary

```
loop until some convergence criteria is met:
    loop through $i = 1, \ldots, d$:
        update $\mathbf{x}_i$ to minimize $f$ with respect to $\mathbf{x}_i$.
```

We call the minimization with respect to one block of coordinates a **micro-step** and one loop through i a **pass**. This algorithm is the simplest form of block coordinate descent (BCD) (see e.g. [52, 23, 48, 2, 41, 53] and many textbooks). When the blocks consist of single coordinates, BCD is called alternating coordinate [22] or coordinate descent (CD) (e.g. [23, 30]).

Despite (or perhaps due to) their simplicity, BCD methods are widely used and promising for many applications such as high-dimensional data analysis [30], machine learning [36, 24], image processing [53] and others [15, 49]. In the context of low rank tensor approximation problems, BCD is known as alternating least-squares (ALS) (e.g. [38, 32, 3, 4, 50, 27, 29, 19, 20, 7, 16, 6, 18, 45, 51, 40, 17, 46]). A micro-step of ALS reduces to a linear least squares problem and so is extremely efficient and precise. Overall, ALS is observed to converge rapidly in many cases. In other cases, however, it exhibits long periods of very slow progress, in what is informally called a **swamp**. Swamps can be classified as **terminal** or **transient**. In a terminal swamp, the slow progress continues to a (local) minimum point. In a transient swamp, progress eventually accelerates and the iterates exit the swamp. Analysis of the causes of swamps can be broken into two questions:

1. What features of a generic f can cause BCD algorithms to progress slowly?
2. What aspects of the tensor approximation problem cause such features to occur so frequently and strongly?

Narrow valleys (thin ridges in maximization problems) have been recognized as a challenge for optimization algorithms for many years (see [34, 35, 42, 11, 14]) and one of the classical test functions for optimization, the Rosenbrock function [42], has a minimum in a narrow valley that is diagonally oriented at the minimum. Thus, narrow valleys are a natural candidate as a cause for swamps and answer for Question 1. In [13] we developed a rudimentary quantitative theory for measuring the effect of narrow valleys on algorithms, based on the gradient descent with line search (GDLS) algorithm. For BCD methods, the orientation of the valley is important. In Figure 1 we illustrate iterations of a CD method in transient and terminal valleys in $\mathbb{R}^2$. Roughly speaking, the problem is that iterations of a BCD method zig-zag slowly in a narrow valley. It is also clear that a valley will attract a non-trivial open set of points under a BCD method.

For the tensor approximation problem, in [13] we found that nonhyperbolic sinks and saddles can occur for certain parameters values and these cause swamps. For nearby parameter values the weakly hyperbolic sinks and saddles create narrow valleys that also cause swamps. See Figure 2 for an illustration of such narrow valleys. In [28] we found that the tensor approximation problem often and robustly contains saddle-like essential (non-removable) discontinuities such as the one plotted in Figure 3. Such discontinuities occur on the boundary of the low rank approximation problem and persist under perturbations of the target tensor. Extremely narrow valleys emanate from these discontinuities and it was observed that large sets of initial conditions can be attracted to a neighborhood of these saddle-like essential discontinuities. Orbits leave the neighborhood along the narrow valleys. This feature may explain the frequent and robust appearance of transient swamps in tensor approximation problems.

In the current paper, we take the results in [13, 28] to be a sufficient answer for Question 2. Thus we take the tensor approximation problem and ALS as motivation, but turn our attention back to

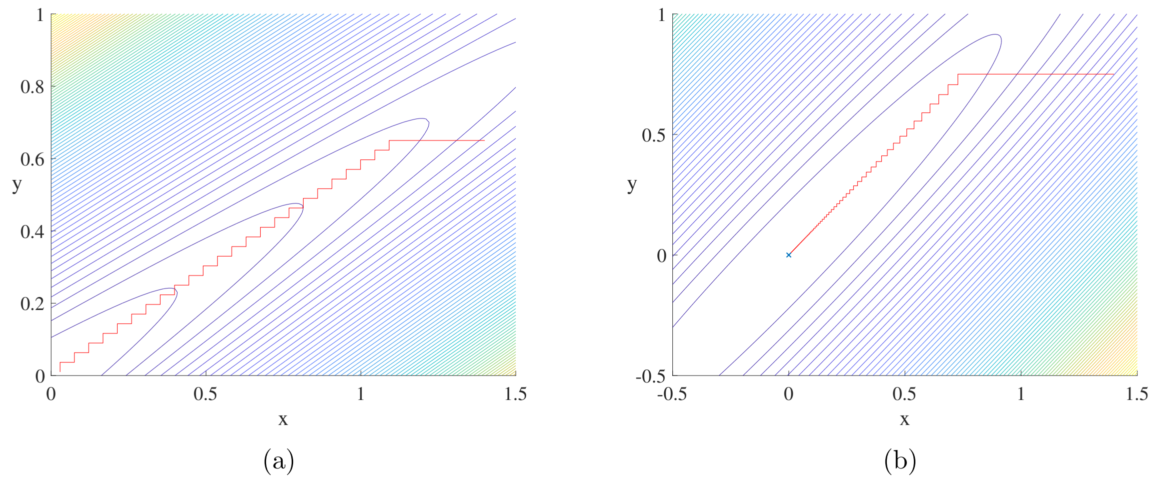


Figure 1: Illustrations of the behavior of a coordinate descent (CD) method in valleys in $\mathbb{R}^2$. The first panel shows iterations along a straight diagonal valley far from any local minimum and the valley floor is at an angle $\pi/6$ from the x-axis. In the second panel iterations are shown near a local minimum point that is in a diagonal valley that is $\pi/4$ from the coordinate directions. Contour curves are shown in both panels. In both cases the trajectories zig-zag slowly downhill along the valley floor.

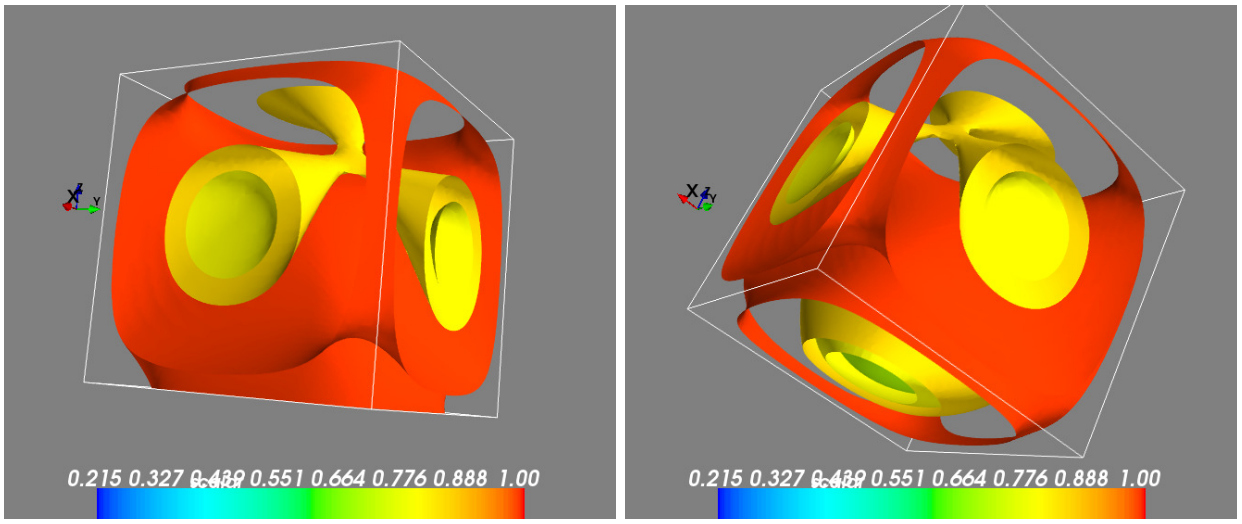


Figure 2: Two views of the level surfaces in the optimization landscape for the problem of fitting a specific $2 \times 2 \times 2$ tensor of rank 2 by a tensor of rank 1. The coordinates are angular so opposite faces are identified. Three narrow diagonal valleys (yellow tube-like) can be seen leading to the global minimizer (inside the green).

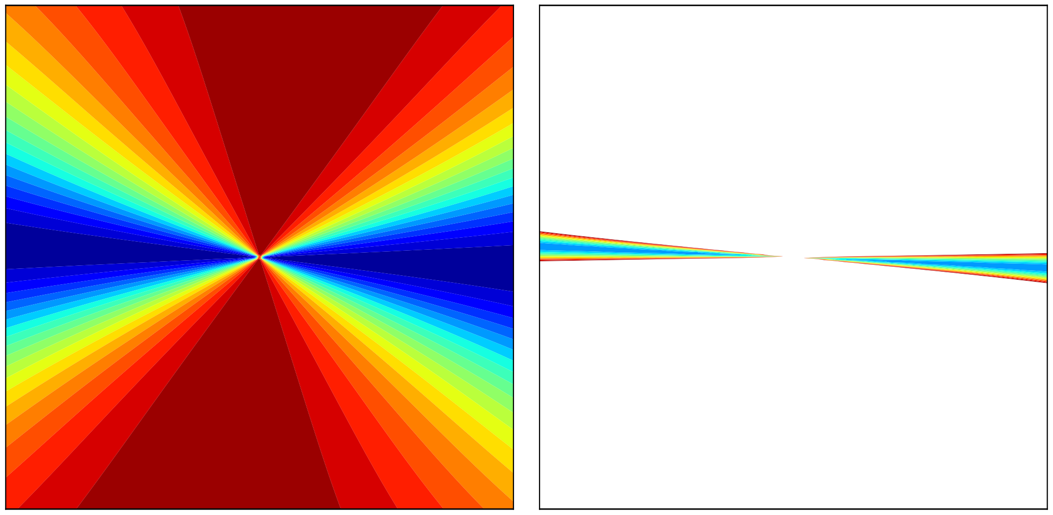


Figure 3: Illustration of a saddle-like essential discontinuity using contour plots with blue small and red large. The left panel is the objective function that appears in the transient swamp in [13, Section 4.2] with contours in $[0, 1]$. The right panel is the same objective function, but with contours in $[0.01, 0.03]$ and white indicating values above 0.03. Such discontinuities seem to exist robustly on the boundary of tensor approximation problems. The objective function is a function of 12 variables in 6 blocks, so only a slice is shown. Iterates drawn near the discontinuity will leave the neighborhood extremely slowly along the narrow valley shown.

Question 1 and BCD in general. Convergence of BCD methods has been studied for many years and with many different assumptions on the objective function and variations on the algorithm (e.g. [52, 23, 48, 30, 2, 39, 53, 41]). Local convergence for ALS was studied in [50]. Here we instead seek to understand transient behavior, specifically:

- How does the progress of BCD depend on the gradient down the valley floor and the narrowness of the valley?
- How does the progress of BCD depend on the direction of the valley with respect to the partitioning of coordinates into blocks?
- How fast do the BCD iterates converge to a vicinity of the valley floor?
- How does asymmetry in the narrowness of the valley affect BCD?

In order to focus on these transient behaviors, we consider an infinitely long, straight valley. One can then use the results as a local model for behavior in a valley that does include a minimum point but where the iterates are currently not near the minimum.

In Section 2 we present our main analysis of BCD dynamics in a valley given by a locally quadratic normal form. We use $\mathbf{v}$ to denote the uphill direction along the bottom of the valley floor. The partition of coordinates $\mathbf{x} = (\mathbf{x}_1; \mathbf{x}_2; \ldots; \mathbf{x}_d)$ induces a corresponding partition of the direction vector $\mathbf{v} = (\mathbf{v}_1; \mathbf{v}_2; \ldots; \mathbf{v}_d)$ that defines the valley floor. We will say that the valley is **maximally diagonal** in the case when $|\mathbf{v}_1| = |\mathbf{v}_2| = \cdots = |\mathbf{v}_d| = 1/\sqrt{d}$. We use ϵ to parameterize the magnitude of the gradient on the valley floor. In Section 2.2 we derive the effect of the BCD algorithm applied to a valley, show that iterates converge linearly to a vicinity of the valley floor, and show that progress in the direction $-\mathbf{v}$ is proportional to ϵ.

In Section 2.3 we assume that the valley sides are symmetric about the valley floor and use σ to parameterize the magnitude of the eigenvalue of the Hessian restricted to the hyperplane orthogonal to $\mathbf{v}$. Under the assumption of symmetric sides, we find the following.

- The rate of progress is proportional to ϵ/σ. Thus, as expected, narrower valleys cause slower progress.
- For any $d \geq 2$, the overall rate of progress down the valley depends on the direction as
$$\left(\sum_{i=1}^{d} \sum_{j=1}^{i-1} |\mathbf{v}_i|^2 |\mathbf{v}_j|^2 \right)^{-1} .$$
  Consequently:
  - The overall rate of progress down the valley is slowest when the valley direction is maximally diagonal.
  - Unless the valley direction $\mathbf{v}$ happens to be very close to the span of *one* of the d blocks, the progress will be almost as slow as for a maximally diagonal direction. In other words, BCD behaves badly for most directions $\mathbf{v}$.
- The iterations converge linearly to a vicinity of the valley floor and then zig-zag at a distance $\sim \epsilon/\sigma$ from the valley floor. For $d = 2$ and $d = 3$ we prove that the rate of convergence toward

the vicinity of the valley floor is at most 1/8 per pass. We show numerically that the rate of convergence is less than 1/8 in the maximally diagonal case for $3 < d \leq 100$. We conjecture that the rate is at most 1/8 for any symmetric valley.

- The order in which blocks of coordinates are updated can significantly affect the total number of iterations needed. In $d = 2$, choosing the block that reduces $f(\mathbf{x})$ the most will also give the most progress down the valley, but for $d > 2$ that is no longer true.

In Section 2.4 we remove the assumption of symmetric valley sides but add the assumption that each block is size one (so BCD reduces to CD). With these assumptions, we show the following.

- When $\mathbf{v}$ is maximally diagonal, progress is inversely proportional to the trace of the Hessian restricted to the hyperplane orthogonal to $\mathbf{v}$. When $\mathbf{v}$ is not maximally diagonal, progress is inversely proportional to a weighted sum of the eigenvalues, but not the trace.
- Iterations converge linearly to a neighborhood of the valley floor, but there does not exist a uniform bound on the rate of convergence, as there was in the symmetric case. This rate may approach 1 from below in some limits.

In Section 3 we briefly consider the dynamics at valley-like sinks and saddles in order to relate them to the dynamics in a valley. We consider a hyperbolic sink or saddle, a nonhyperbolic sink or saddle, and an essential discontinuity that is sink-like or saddle-like. Now using ϵ as the strength of the attraction or repulsion in the weak direction, we find progress is again proportional to ϵ/σ. In the hyperbolic case we find that progress is proportional to the current distance to the sink or saddle. In the nonhyperbolic and essential discontinuity cases we find that the progress is proportional to the cube of this distance, and thus is much slower.

In Section 4, we discuss some implications of our analysis on the use of BCD and ALS methods and potential improvements in their performance.

2 BCD Dynamics in a Valley

We will use the convention that $\mathbf{x} \in \mathbb{R}^n$ is a column vector and $\mathbf{x}^T$ is its transpose. The usual inner (dot) product of $\mathbf{x}$ and $\mathbf{y}$ is then $\mathbf{x}^T\mathbf{y}$ and the usual 2-norm is $|\mathbf{x}| = \sqrt{\mathbf{x}^T\mathbf{x}}$. Elements gathered into a row vector are written $(x_1, x_2, \dots)$ whereas elements gathered into a column are written $(x_1; x_2; \dots)$; the latter notation allows us to gather column vectors into a column vector as in $\mathbf{x} = (\mathbf{x}_1; \mathbf{x}_2; \dots; \mathbf{x}_d)$.

2.1 The Model Problem

Near some point $\mathbf{x}_0$, the locally quadratic "normal form" of f is given by

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^T(\mathbf{x} - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^T H(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0)\,,$$

where $H(\mathbf{x}_0)$ is the Hessian of f at $\mathbf{x}_0$. The point $\mathbf{x}_0$ will be a point of interest, such as a sink or saddle or point on the floor of a valley of the objective function f. In studying local dynamics, without loss of generality one can take $\mathbf{x}_0 = \mathbf{0}$ and $f(\mathbf{x}_0) = 0$. The features we wish to study have a primary orientation, which in the case of a narrow valley is the direction of $\nabla f(\mathbf{x}_0)$. We will assume $|\nabla f(\mathbf{x}_0)|$ is small, so we will replace $\nabla f(\mathbf{x}_0)$ by $\epsilon\mathbf{v}$, where $\mathbf{v}$ is a unit vector and $\epsilon \geq 0$. The Hessian has effects both along $\mathbf{v}$ and orthogonal to $\mathbf{v}$. We assume these split cleanly in that $\mathbf{v}$ is an eigenvector of $H(\mathbf{x}_0)$. The features also attract the gradient flow to $V = \text{span}(\mathbf{v})$, so we assume the portion of $H(\mathbf{x}_0)$ orthogonal to $\mathbf{v}$ is positive definite.

For a valley, we therefore consider the form

$$f(\mathbf{x}) = \epsilon\mathbf{v}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T H^{\perp}\mathbf{x}\,, \tag{1}$$

where $H^{\perp}$ is a positive semi-definite matrix with nullspace V and $\epsilon > 0$. When the nonzero eigenvalues of $H^{\perp}$ are large compared to ϵ, then the valley is narrow. The gradient of f is

$$\nabla f(\mathbf{x}) = \epsilon\mathbf{v} + H^{\perp}\mathbf{x}\,. \tag{2}$$

2.2 Block Coordinate Descent

Suppose now we partition $\mathbb{R}^n$ into $d > 1$ sets of variables, i.e. $\mathbf{x} = (\mathbf{x}_1; \mathbf{x}_2; \ldots; \mathbf{x}_d)$. This induces a partition $\mathbf{v} = (\mathbf{v}_1; \mathbf{v}_2; \ldots; \mathbf{v}_d)$ and a partition of $H^{\perp}$ into blocks $H^{\perp}_{ij}$. We assume always that $|\mathbf{v}_i| > 0$ for all i. By a BCD method, we will mean minimizing with respect to the coordinate blocks $\mathbf{x}_i$ sequentially and cyclically. To minimize in $\mathbf{x}_i$, one sets the partial gradient with respect to $\mathbf{x}_i$ equal to zero and solves for $\mathbf{x}_i$. From (2) we obtain

$$\nabla_{\mathbf{x}_i} f(\mathbf{x}) = \epsilon\mathbf{v}_i + \sum_{j=1}^{d} H^{\perp}_{ij}\mathbf{x}_j = \epsilon\mathbf{v}_i + H^{\perp}_{ii}\mathbf{x}_i + \sum_{j=1,\neq i}^{d} H^{\perp}_{ij}\mathbf{x}_j\,.$$

We can solve $\nabla_{\mathbf{x}_i} f(\mathbf{x}) = \mathbf{0}$ for $\mathbf{x}_i$ to obtain

$$\mathbf{x}_i^{\text{new}} = -\left(H^{\perp}_{ii}\right)^{-1}\left(\sum_{j=1,\neq i}^{d} H^{\perp}_{ij}\mathbf{x}_j + \epsilon\mathbf{v}_i\right)\,. \tag{3}$$

If $H^{\perp}_{ii}$ were not invertible, then there would be a non-zero $\mathbf{w}$ such that $H^{\perp}_{ii}\mathbf{w} = \mathbf{0}$. Setting $\mathbf{w}' = (\mathbf{0}; \ldots; \mathbf{0}; \mathbf{w}; \mathbf{0}; \ldots; \mathbf{0})$, i.e. the zero vector except $\mathbf{w}$ in the i-th block, we would obtain $\mathbf{w}'^T H^{\perp}\mathbf{w}' = 0$. We assumed $H^{\perp}$ was positive semi-definite with nullspace V, so $\mathbf{w}'$ must be a multiple of $\mathbf{v}$. We also assumed $|\mathbf{v}_j| > 0$ for all j, which does not allow $\mathbf{w}'$ to have zero blocks. Thus $H^{\perp}_{ii}$ is invertible.

We can also write (3) as an update of the whole vector $\mathbf{x}$ as

$$\mathbf{x}^{\text{new}} = M_i\mathbf{x} - \mathbf{a}_i\,, \tag{4}$$

where M_i is an $n \times n$ matrix and $\mathbf{a}_i$ a length n vector corresponding to the update of the ith block of $\mathbf{x}$. Both M_i and $\mathbf{a}_i$ are partitioned into blocks, which are given by

$$(M_i)_{kj} = \begin{cases} 0 & \text{if } k \neq i \text{ and } j \neq k \\ I & \text{if } k \neq i \text{ and } j = k \\ 0 & \text{if } k = i \text{ and } j = k \\ -\left(H^{\perp}_{ii}\right)^{-1} H^{\perp}_{ij} & \text{if } k = i \text{ and } j \neq i \end{cases} \quad \text{and} \tag{5}$$

$$(\mathbf{a}_i)_k = \begin{cases} 0 & \text{if } k \neq i \\ -\left(H^{\perp}_{ii}\right)^{-1}\epsilon\mathbf{v}_i & \text{if } k = i \end{cases}\,. \tag{6}$$

A BCD pass through the directions can be written recursively as

$$\mathbf{x}^{\text{new}} = M_d\left(\cdots\left(M_2\left(M_1\mathbf{x}+\mathbf{a}_1\right)+\mathbf{a}_2\right)+\cdots\right)+\mathbf{a}_d\,.$$

By collecting the terms involving $\mathbf{x}$, we can write this as

$$\begin{aligned}\mathbf{x}^{\text{new}} &= M\mathbf{x}+\mathbf{b} \quad \text{with}\\ M &= M_d\cdots M_2M_1 \quad \text{and} \qquad (7)\\ \mathbf{b} &= M_d\left(\cdots\left(\left(M_2\mathbf{a}_1\right)+\mathbf{a}_2\right)+\cdots\right)+\mathbf{a}_d = \sum_{i=1}^{d} M_d\cdots M_{i+1}\mathbf{a}_i\,. \qquad (8)\end{aligned}$$

The effect of applying M is to move $\mathbf{x}$ toward $\mathbf{v}$. We can formalize this in terms of its eigenvalues and eigenvectors.

Theorem 2.1. *The matrix M in (7) has an eigenvalue 1 with eigenvector $\mathbf{v}$; all its other eigenvalues satisfy $|\lambda| < 1$.*

Proof. If we apply M_i defined by (5) to $\mathbf{v}$, then we have $(M_i\mathbf{v})_k = I\mathbf{v}_k$ for $k \neq i$. Since $H^\perp\mathbf{v} = \mathbf{0}$, we have $\sum_{j=1,\neq i}^{d} H^\perp_{ij}\mathbf{v}_j = -H^\perp_{ii}\mathbf{v}_i$, so the $k=i$ block gives

$$(M_i\mathbf{v})_i = -\left(H^\perp_{ii}\right)^{-1}\sum_{j=1,\neq i}^{d} H^\perp_{ij}\mathbf{v}_j = -\left(H^\perp_{ii}\right)^{-1}\left(-H^\perp_{ii}\mathbf{v}_i\right) = \mathbf{v}_i\,.$$

Thus $\mathbf{v}$ is an eigenvector with eigenvalue 1 for each M_i and hence also for their product M.

To analyze the remaining eigenvalues, consider the BCD update process when $\epsilon = 0$ and hence $\mathbf{a}_i = \mathbf{b} = \mathbf{0}$. Each step computes $\nabla_{\mathbf{x}_i}\frac{1}{2}\mathbf{x}^TH^\perp\mathbf{x}$ and, if this is nonzero, updates $\mathbf{x}$ to reduce $\frac{1}{2}\mathbf{x}^TH^\perp\mathbf{x}$. Since $H^\perp$ is positive semi-definite, we always have $0 \leq \frac{1}{2}\mathbf{x}^TH^\perp\mathbf{x}$. Thus, for any $\mathbf{x}$, either $0 \leq (M\mathbf{x})^TH^\perp M\mathbf{x} < \mathbf{x}^TH^\perp\mathbf{x}$ or $\nabla\frac{1}{2}\mathbf{x}^TH^\perp\mathbf{x} = \mathbf{0}$. Since $\nabla\frac{1}{2}\mathbf{x}^TH^\perp\mathbf{x} = H^\perp\mathbf{x}$, the second case implies $\mathbf{x}$ is in the nullspace of $H^\perp$, which means it is a multiple of $\mathbf{v}$.

Suppose $\mathbf{w}$ is an eigenvector of M with eigenvalue λ, and $\mathbf{w}$ is not a multiple of $\mathbf{v}$. If λ is real, then $\mathbf{w}$ can also be taken real and we have

$$0 \leq (M\mathbf{w})^TH^\perp M\mathbf{w} = \lambda^2\mathbf{w}^TH^\perp\mathbf{w} < \mathbf{w}^TH^\perp\mathbf{w}$$

so $|\lambda| < 1$. If λ is complex, then its complex conjugate $\overline{\lambda}$ is an eigenvalue with eigenvector $\overline{\mathbf{w}}$. Since $\mathbf{w}+\overline{\mathbf{w}}$ and $i(\mathbf{w}-\overline{\mathbf{w}})$ are real, we have

$$\begin{aligned}&0 \leq (\lambda\mathbf{w}+\overline{\lambda}\overline{\mathbf{w}})^TH^\perp(\lambda\mathbf{w}+\overline{\lambda}\overline{\mathbf{w}}) < (\mathbf{w}+\overline{\mathbf{w}})^TH^\perp(\mathbf{w}+\overline{\mathbf{w}}) \quad \text{and}\\ &0 \leq -(\lambda\mathbf{w}-\overline{\lambda}\overline{\mathbf{w}})^TH^\perp(\lambda\mathbf{w}-\overline{\lambda}\overline{\mathbf{w}}) < -(\mathbf{w}-\overline{\mathbf{w}})^TH^\perp(\mathbf{w}-\overline{\mathbf{w}})\,.\end{aligned}$$

Adding these inequalities yields

$$0 \leq 2|\lambda|^2\left(\mathbf{w}^TH^\perp\overline{\mathbf{w}}+\overline{\mathbf{w}^TH^\perp\overline{\mathbf{w}}}\right) < 2\left(\mathbf{w}^TH^\perp\overline{\mathbf{w}}+\overline{\mathbf{w}^TH^\perp\overline{\mathbf{w}}}\right)\,,$$

so $|\lambda| < 1$. □

Assume for the moment that the eigenvalues of M are distinct. Let Λ be the set of eigenvalues of M and $\mathbf{w}_\lambda$ an eigenvector for eigenvalue λ. Let $\mathbf{b} = \sum_{\lambda\in\Lambda} \hat{b}_\lambda \mathbf{w}_\lambda$ and $\mathbf{x}^0 = \sum_{\lambda\in\Lambda} \hat{x}_\lambda \mathbf{w}_\lambda$ be the expressions of $\mathbf{b}$ and the initial condition $\mathbf{x}^0$ in the basis of eigenvectors. Then, for $k > 0$, solutions of the recurrence satisfy

$$\mathbf{x}^k = M^k\mathbf{x}^0 + \sum_{i=0}^{k-1} M^n\mathbf{b} = \sum_{\lambda\in\Lambda}\left(\lambda^k\hat{x}_\lambda + \sum_{i=0}^{k-1}\lambda^i\hat{b}_\lambda\right)\mathbf{w}_\lambda\,. \tag{9}$$

The term with $\lambda = 1$ contributes $(\hat{x}_1 + k\hat{b}_1)\mathbf{w}_1$, which progresses along $\mathbf{w}_1$ with rate $\hat{b}_1$. For each $|\lambda| < 1$, the term contributes $(\lambda^k\hat{x}_\lambda + (1-\lambda^k)(1-\lambda)^{-1}\hat{b}_\lambda)\mathbf{w}_\lambda$, which converges to $(1-\lambda)^{-1}\hat{b}_\lambda\mathbf{w}_\lambda$ with rate $|\lambda|$. Observe that $\lambda \approx 1$ causes large separation from $\mathbf{w}_1$ as well as slow convergence while $|\lambda| \approx 1$ only causes slow convergence. We have shown in Theorem 2.1 that M has a simple eigenvalue 1 with eigenvector $\mathbf{w}_1 = \mathbf{v}$ and that all other eigenvalues satisfy $|\lambda| < 1$. Thus iterations will converge linearly to

$$\mathbf{x}^k \approx (\hat{x}_1 + k\hat{b}_1)\mathbf{v} + \sum_{|\lambda|<1}(1-\lambda)^{-1}\hat{b}_\lambda\mathbf{w}_\lambda.$$

If the eigenvalues of M are not distinct but it is still diagonalizable, then a basis of eigenvectors still exist. The above analysis still holds, but the notation becomes cumbersome.

If M is not diagonalizable, then some eigenvalues have geometric multiplicity less than their algebraic multiplicity. Let λ be such an eigenvalue, which we now fix so that we can suppress it from the notation. By Theorem 2.1, we know $|\lambda| < 1$. Corresponding to each $m \times m$ block associated to λ in the Jordan canonical form of M there is a chain of m linearly independent vectors $\{\mathbf{y}_j\}_{j=1}^m$ such that $(A - \lambda I)\mathbf{y}_j = \mathbf{y}_{j-1}$ and $(A - \lambda I)\mathbf{y}_1 = \mathbf{0}$. The generalized eigenvectors $\{\mathbf{y}_j\}_{j=2}^m$ make up for the lack of enough eigenvectors to form a basis; note that $\mathbf{y}_1 = \mathbf{w}_\lambda$ is an eigenvector and so is included in the previous analysis. We can compute directly that

$$M^k\mathbf{y}_j = \sum_{q=0}^{j-1}\binom{k}{q}\lambda^{k-q}\mathbf{y}_{j-q} \quad \text{for } k \geq m \text{ and} \tag{10}$$

$$\lim_{k\to\infty}\sum_{i=0}^{k-1} M^i\mathbf{y}_j = \sum_{q=0}^{j-1}\frac{1}{(1-\lambda)^{q+1}}\mathbf{y}_{j-q}\,. \tag{11}$$

Since $|\lambda| < 1$, the terms (10) converge to zero linearly with rate of convergence μ for any μ with $|\lambda| < \mu < 1$. The expansions of $\mathbf{x}^0$ and $\mathbf{b}$ will include terms corresponding to $\{\mathbf{y}_j\}_{j=2}^m$. In (9) we will then have have a coefficient of $\mathbf{x}^0$ times (10), which still converges to zero linearly. Similarly, we will have a coefficient of $\mathbf{b}$ times $\sum_{i=0}^{k-1} M^i\mathbf{y}_j$, which converges linearly to (11).

Our first goal in subsequent sections will be to find $\hat{b}_1$ to determine the progress rate down the valley for various cases. We can already observe that $\hat{b}_1$ is proportional to ϵ since $\mathbf{b}$ contains a factor of ϵ coming from $\mathbf{a}_i$ in (6). Our second goal will be to find the maximum of $\{|\lambda| : |\lambda| < 1\}$ to determine the convergence rate toward a vicinity of the valley floor.

2.3 Dynamics in a Symmetric Valley

2.3.1 Formulation and Reduction to Coefficients

When the attraction to V is radially symmetric about V, then for some $\sigma > 0$ we can write

$$\begin{aligned} H^{\perp} &= \sigma(I - \mathbf{v}\mathbf{v}^T), \quad \text{and thus have} \qquad (12)\\ f(\mathbf{x}) &= \epsilon \mathbf{v}^T\mathbf{x} + \frac{\sigma}{2}(\mathbf{x}^T\mathbf{x} - (\mathbf{v}^T\mathbf{x})^2) \quad \text{and} \\ \nabla f(\mathbf{x}) &= \epsilon\mathbf{v} + \sigma(\mathbf{x} - (\mathbf{v}^T\mathbf{x})\mathbf{v})\,. \end{aligned}$$

Note that $\mathbf{x}^T\mathbf{x} - (\mathbf{v}^T\mathbf{x})^2$ is the distance from $\mathbf{x}$ to V. The update (3) becomes

$$\begin{aligned} \mathbf{x}_i &= \left(\sigma(I - \mathbf{v}_i\mathbf{v}_i^T)\right)^{-1}\left(\sigma \sum_{j=1,\neq i}^{d} \mathbf{v}_i\mathbf{v}_j^T\mathbf{x}_j - \epsilon\mathbf{v}_i\right) \\ &= \left(\sigma \sum_{j=1,\neq i}^{d} \mathbf{v}_j^T\mathbf{x}_j - \epsilon\right)\left(\sigma I - \sigma\mathbf{v}_i\mathbf{v}_i^T\right)^{-1}\mathbf{v}_i \\ &= \left(\sigma \sum_{j=1,\neq i}^{d} \mathbf{v}_j^T\mathbf{x}_j - \epsilon\right)\frac{1}{\sigma(1 - \mathbf{v}_i^T\mathbf{v}_i)}\mathbf{v}_i\,. \qquad (13) \end{aligned}$$

Note that the updated $\mathbf{x}_i$ is a multiple of $\mathbf{v}_i$, so we can write $\mathbf{x}_i = c_i\mathbf{v}_i$. After one pass through the directions, we will have $\mathbf{x}_j = c_j\mathbf{v}_j$ for all j. Thus a pass of BCD maps the entire space onto the subspace $W = (c_1\mathbf{v}_1; c_2\mathbf{v}_2; \ldots; c_d\mathbf{v}_d)$ and W is invariant under micro-steps of BCD. We remark that V is a subspace of W, but V is not invariant under micro-steps or the full BCD algorithm. Note that if $\mathbf{v}_j = \mathbf{0}$ then the update would produce $\mathbf{x}_j = \mathbf{0}$ and c_j then has no meaning or role. We have assumed $|\mathbf{v}_j| > 0$ for all j to avoid such null directions. Let $p_i = \mathbf{v}_i^T\mathbf{v}_i$ and $\mathbf{p}$ be the vector of p_i values. We can then exchange the vector update (13) for the scalar update

$$c_i = \left(\sigma \sum_{j=1,\neq i}^{d} c_j p_j - \epsilon\right)\frac{1}{\sigma(1 - p_i)} = \frac{1}{1 - p_i}\left(\sum_{j=1,\neq i}^{d} c_j p_j - \frac{\epsilon}{\sigma}\right)\,. \qquad (14)$$

In what follows we assume there has already been one update pass through the directions, so $\mathbf{x} \in W$ and we can work entirely with the coefficient vector $\mathbf{c} = (c_1; \ldots; c_d)$, which we assume for convenience has initial value $\mathbf{c}^0 = (c_1^0; \ldots; c_d^0)$.

Since $|\mathbf{v}|^2 = \sum_{j=1}^{d} p_j = 1$, for interpretation we can rewrite (14) as

$$c_i = \frac{\sum_{j=1,\neq i}^{d} c_j p_j}{\sum_{j=1,\neq i}^{d} p_j} - \frac{\epsilon}{\sigma(1 - p_i)}\,. \qquad (15)$$

The first term is a weighted average of $\{c_j\}_{j\neq i}$ and so tries to make all $\{c_j\}_{j=1}^{d}$ the same. The second term controls the drift down the valley. It is proportional to ϵ, which is the size of the gradient on the valley floor. It is inversely proportional to σ, so the narrower the valley is, the slower the progress down the valley. It is also inversely proportional to $(1 - p_i) \in (0, 1)$, so the update of the i-block achieves more progress if $\mathbf{v}$ has a larger component in $\mathbf{v}_i$.

We can also write (14) as an update of the whole vector $\mathbf{c}$ as

$$\mathbf{c}^{\text{new}} = M_i\mathbf{c} - \frac{\epsilon}{\sigma(1-p_i)}\mathbf{e}_i \quad \text{with} \tag{16}$$

$$M_i = \left(I + \frac{1}{1-p_i}\mathbf{e}_i\left(\mathbf{p}-\mathbf{e}_i\right)^T\right)\,. \tag{17}$$

We note that M and M_i here are matrices representing the same linear operators as in section 2.2, but restricted to the subspace W. A BCD pass through the directions can then be written

$$\begin{aligned}\mathbf{c}^{\text{new}} &= M\mathbf{c} + \mathbf{b} \quad \text{with}\\ M &= M_d \cdots M_2 M_1 \quad \text{and} \end{aligned}\tag{18}$$

$$\mathbf{b} = \frac{-\epsilon}{\sigma}\sum_{i=1}^{d} M_d \cdots M_{i+1}\frac{1}{1-p_i}\mathbf{e}_i\,. \tag{19}$$

Corresponding to Theorem 2.1. we have

Corollary 2.2. *The matrix M in (18) has an eigenvalue 1 with eigenvector $\mathbf{1}$; all its other eigenvalues satisfy $|\lambda| < 1$.*

Proof. By direct computation, $M_i\mathbf{1} = \mathbf{1}$ for all i, so M has an eigenvalue 1 with eigenvector $\mathbf{1}$. The rest of the theorem follows from Theorem 2.1 since the current M and M_i are simply restrictions of the linear operators in Section 2.2 to V. □

The convergence theory in Section 2.2 still applies, now replacing $\mathbf{x}$ by $\mathbf{c}$ and $\mathbf{v}$ by $\mathbf{1}$. We are thus still interested in finding $\hat{b}_1$ and the maximum of $\{|\lambda| : |\lambda| < 1\}$. Note from (19) that $\mathbf{b}$ has a factor ϵ/σ and therefore so does each $\hat{b}_\lambda$. Thus the progress along the valley and the asymptotic distance from the floor of the valley are both proportional to ϵ/σ.

2.3.2 The case $d = 2$

When $d = 2$ we have $p_1 + p_2 = 1$ and the update (14) yields

$$\begin{aligned} c_1^1 &= \frac{1}{1-p_1}\left(c_2^0 p_2 - \frac{\epsilon}{\sigma}\right) = c_2^0 - \frac{1}{p_2}\frac{\epsilon}{\sigma} \quad \text{and}\\ c_2^1 &= \frac{1}{1-p_2}\left(c_1^1 p_1 - \frac{\epsilon}{\sigma}\right) = c_1^1 - \frac{1}{p_1}\frac{\epsilon}{\sigma} = c_2^0 - \frac{1}{p_1 p_2}\frac{\epsilon}{\sigma}\,. \end{aligned}$$

Continuing, we have

$$\begin{aligned} c_1^k &= c_2^0 - \left(\frac{k-1}{p_1p_2} + \frac{1}{p_2}\right)\frac{\epsilon}{\sigma} = c_2^0 - \left(\frac{k}{p_1p_2} - \frac{1}{p_1}\right)\frac{\epsilon}{\sigma} \quad \text{and}\\ c_2^k &= c_2^0 - \frac{k}{p_1p_2}\frac{\epsilon}{\sigma}\,. \end{aligned}$$

The coefficients produced by the steps of the algorithm alternate between two parallel lines. Namely, (c_1^k, c_2^{k-1}) is always on the line $c_2 = c_1 + \frac{\epsilon}{p_2\sigma}$ and (c_1^k, c_2^k) is on the line $c_2 = c_1 - \frac{\epsilon}{p_1\sigma}$. Therefore, iterates in the original coordinates alternate between the two parallel lines as

$$\begin{pmatrix}\mathbf{x}_1^k\\ \mathbf{x}_2^{k-1}\end{pmatrix} \in V + \frac{\epsilon}{p_2\sigma}\begin{pmatrix}\mathbf{0}\\ \mathbf{v}_2\end{pmatrix} \qquad \text{and} \qquad \begin{pmatrix}\mathbf{x}_1^k\\ \mathbf{x}_2^k\end{pmatrix} \in V - \frac{\epsilon}{p_1\sigma}\begin{pmatrix}\mathbf{0}\\ \mathbf{v}_2\end{pmatrix}\,.$$

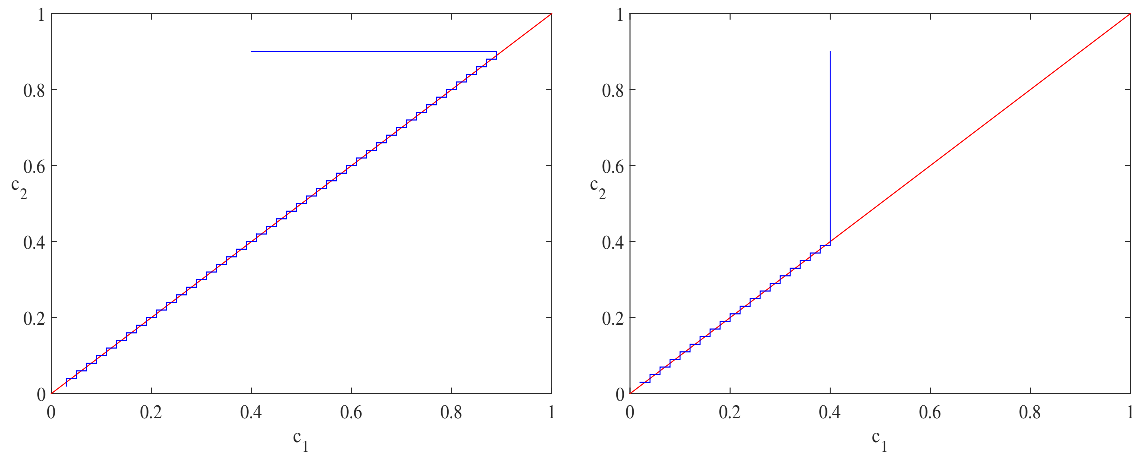


Figure 4: Trajectory of the coefficients (c_1, c_2) generated by BCD in a symmetric valley with $d = 2$. In the left panel c_1 is optimized first and in the right panel c_2 is optimized first. In both the trajectory is immediately projected onto a pair of parallel lines close to $\{c_1 = c_2\}$. This illustrates very clearly that the initial order of optimization micro-steps may matter greatly. For two coordinate blocks ($d = 2$) progress is optimized if one chooses to first minimize with respect to the block ($\mathbf{x}_2$ in this example) that gives the greatest improvement in the objective function (i.e. a greedy strategy). We will see in Section 2.3.3 that this does not work for more than two blocks.

These lines are each a distance $\sim \epsilon/\sigma$ from V. In Figure 4 we illustrate the trajectories of coefficients (c_1, c_2).

We can express the direction of $\mathbf{v}$ with respect to its partition into $\mathbf{v}_1$ and $\mathbf{v}_2$ using the angle θ such that $(\cos(\theta), \sin(\theta)) = (\sqrt{p_1}, \sqrt{p_2})$. (Note that θ is the angle between $\mathbf{v}$ and the $\mathbf{x}_1$ coordinate hyperplane.) Each pass of BCD moves $\mathbf{x}$ a distance of

$$|\Delta\mathbf{x}| = \left| \begin{pmatrix} c_1^{k+1}\mathbf{v}_1 \\ c_2^{k+1}\mathbf{v}_2 \end{pmatrix} - \begin{pmatrix} c_1^{k}\mathbf{v}_1 \\ c_2^{k}\mathbf{v}_2 \end{pmatrix} \right| = \frac{1}{p_1 p_2}\frac{\epsilon}{\sigma}|\mathbf{v}| = \frac{1}{\cos^2\theta \sin^2\theta}\frac{\epsilon}{\sigma} = 4\csc^2(2\theta)\frac{\epsilon}{\sigma}\,. \tag{20}$$

The minimal progress of $4\epsilon/\sigma$ occurs when $\theta = \pi/4$, which is the maximally diagonal case $|\mathbf{v}_1| = |\mathbf{v}_2|$. We plot the dependence on θ in Figure 5. For a large domain of angles, the movement is close to the minimum value. In particular, for angles $\pi/8 < \theta < 3\pi/8$, which is half the available domain, the rate is less than twice the minimum value. If we consider for which θ the rate of progress is within an order of magnitude of the minimum value, that occurs for $0.16 < \theta < 1.41$ or about 80% of the domain of angles.

We also observe from the recurrence and Figure 4 that it can matter a lot whether one optimizes first in $\mathbf{x}_1$ or $\mathbf{x}_2$. We see that the difference in the number of iterations can be on the order of σ/ϵ times the distance of $\mathbf{x}^0$ from the valley floor. With two blocks ($d = 2$) it is easy to check that the first micro-step of BCD should be *greedy*. That is, one should calculate the micro-steps with respect to each block of coordinates and update the block that gives the largest decrease in the objective function.

From our investigation of $d = 2$, we draw the following conclusions or lessons:

- Iterations zig-zag along lines $\sim \epsilon/\sigma$ from the valley floor.
- A greedy first micro-step can have a large benefit.

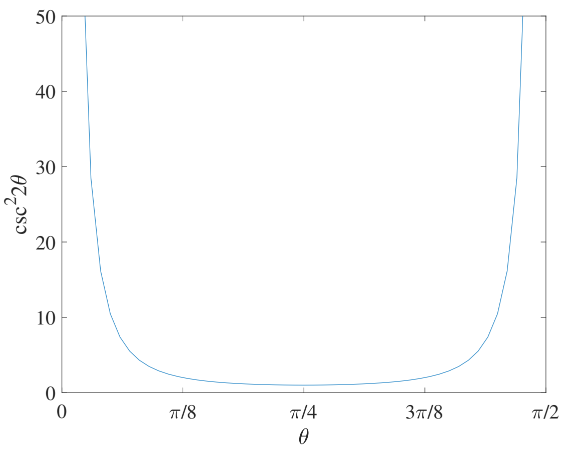


Figure 5: Graph of $\csc^2(2\theta)$, which is proportional to the progress of BCD as a function of the angle of the gradient with respect to two blocks of variables. We see that this in fact is quite small for a large neighborhood of angles around the minimum at $\pi/4$.

- Progress is proportional to ϵ/σ. It depends on the angle θ proportionally to $\csc^2(2\theta)$. The slowest progress rate $4\epsilon/\sigma$ occurs when $|\mathbf{v}_1| = |\mathbf{v}_2|$.

2.3.3 The case $d = 3$

When $d = 3$ we have $p_1 + p_2 + p_3 = 1$ and the update (14) yields

$$\begin{aligned} c_1^{k+1} &= \frac{1}{p_2 + p_3}\left(p_2 c_2^k + p_3 c_3^k - \frac{\epsilon}{\sigma}\right), \\ c_2^{k+1} &= \frac{1}{p_1 + p_3}\left(p_1 c_1^{k+1} + p_3 c_3^k - \frac{\epsilon}{\sigma}\right), \quad \text{and} \\ c_3^{k+1} &= \frac{1}{p_1 + p_2}\left(p_1 c_1^{k+1} + p_2 c_2^{k+1} - \frac{\epsilon}{\sigma}\right). \end{aligned}$$

A trajectory of this recurrence is plotted in Figure 6. Iterations of the coefficients are attracted to a neighborhood of the diagonal $\{c_1 = c_2 = c_3\}$, then zig-zag downward along the valley floor. In particular, in the limit the iterations alternate between three lines that are parallel to the diagonal.

Using (17), we can compute the matrix (18)

$$M = M_3 M_2 M_1 = \begin{pmatrix} 0 & \frac{p_2}{1-p_1} & \frac{p_3}{(1-p_1)} \\ 0 & \frac{p_1 p_2}{(1-p_1)(1-p_2)} & \frac{p_3}{(1-p_1)(1-p_2)} \\ 0 & \frac{p_1 p_2}{(1-p_1)(1-p_2)(1-p_3)} & \frac{(p_1+p_2-p_1p_2)p_3}{(1-p_1)(1-p_2)(1-p_3)} \end{pmatrix}$$

and the vector (19)

$$\mathbf{b} = -\frac{\epsilon}{\sigma}\begin{pmatrix} \frac{1}{1-p_1} \\ \frac{1}{(1-p_1)(1-p_2)} \\ \frac{1}{(1-p_1)(1-p_2)(1-p_3)} \end{pmatrix}.$$

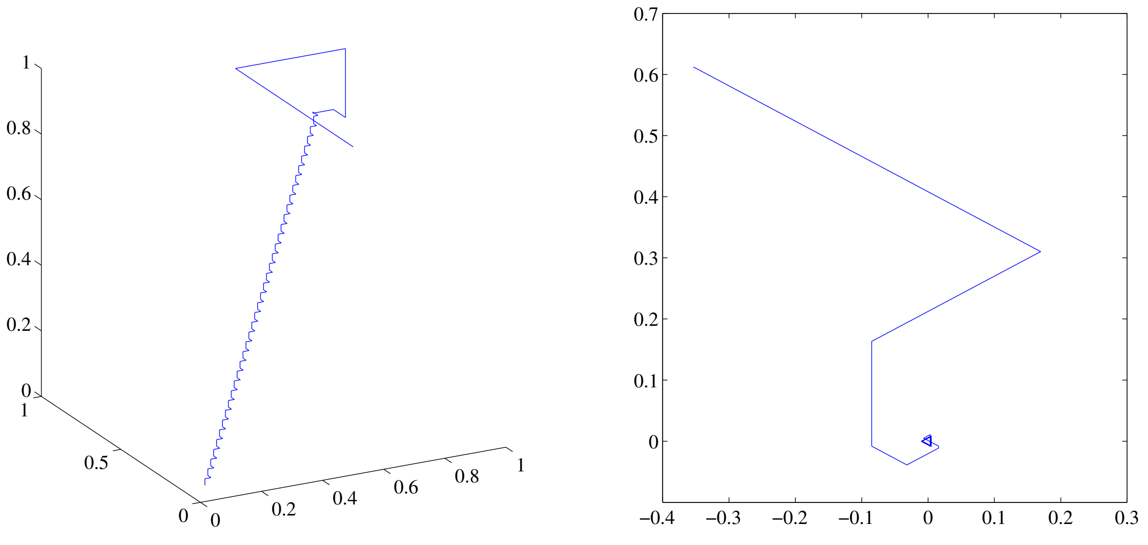


Figure 6: Trajectory of the coefficients (c_1, c_2, c_3) generated by BCD in a symmetric valley with $d = 3$. The first panel shows iterations of the coefficients first approaching a neighborhood of the diagonal $\{c_1 = c_2 = c_3\}$, then moving along three lines that are parallel to the diagonal. In the second panel these iterations are projected onto a plane normal to the diagonal.

The matrix M has an eigenvalue 0 with eigenvector $\mathbf{e}_1$ and (as noted in Corollary 2.2) an eigenvalue 1 with eigenvector $\mathbf{1}$. The third eigenvalue-eigenvector pair is

$$\lambda = \frac{-p_1p_2p_3}{(1-p_1)(1-p_2)(1-p_3)} = \frac{-p_1p_2p_3}{(p_1+p_2)(p_2+p_3)(p_1+p_3)}, \quad \text{and} \tag{21}$$
$$\mathbf{w}_\lambda = \left(\frac{(1-p_2)(1-p_3)}{p_1^2}, \quad \frac{-(1-p_3)p_3}{p_1p_2}, \quad 1 \right)^T .$$

Since $p_i + p_j \geq 2\sqrt{p_ip_j}$, we have that $|\lambda| \leq 1/8$; the case $p_1 = p_2 = p_3 = 1/3$ yields $\lambda = -1/8$ so this bound is sharp.

By solving a linear system, we can express $\mathbf{b}$ in the basis $\{\mathbf{e}_1, \mathbf{1}, \mathbf{w}_\lambda\}$ as $\mathbf{b} = \hat{b}_0\mathbf{e}_1 + \hat{b}_1\mathbf{1} + \hat{b}_\lambda\mathbf{w}$ with

$$\hat{b}_0 = \frac{\epsilon}{\sigma}\frac{1}{p_1}, \qquad \hat{b}_1 = -\frac{\epsilon}{\sigma}\frac{1}{p_1p_2 + p_2p_3 + p_1p_3}, \qquad \text{and}$$
$$\hat{b}_\lambda = -\frac{\epsilon}{\sigma}\frac{p_1p_2p_3}{(1-p_1)(1-p_2)(1-p_3)(p_1p_2+p_2p_3+p_1p_3)}.$$

Similarly, for any starting point $\mathbf{c}^0 \in \mathbf{R}^3$ we have $\mathbf{c}^0 = \hat{c}_0\mathbf{e}_1 + \hat{c}_1\mathbf{1} + \hat{c}_\lambda\mathbf{w}_\lambda$. The expression (9) for the solutions of the recurrence becomes

$$\mathbf{c}^k = \left(\hat{c}_1 - \hat{b}_1 k\right)\mathbf{1} + \left(\hat{c}_\lambda\lambda^k - \hat{b}_\lambda\frac{1-\lambda^k}{1-\lambda}\right)\mathbf{w}_\lambda\,.$$

We see that the coefficients of the recurrence converge to a neighborhood of the diagonal $\{c_1 = c_2 = c_3\}$ (rather than being sent there immediately as in the case $d = 2$) and so iterates of the algorithm converge to a neighborhood of the line spanned by $\mathbf{v}$. The rate of convergence to the valley floor is controlled by the eigenvalue λ in (21). We saw that $|\lambda| \le 1/8$, with equality attained for the symmetric case $p_1 = p_2 = p_3 = 1/3$. Therefore the slowest convergence to the valley is a rate of $1/8$ per pass of the algorithm. The minimum of $|\lambda| = 0$ occurs along the boundary, where the problem reduced to the case $d = 2$. It is of particular note that the rate $|\lambda| \le 1/8$ is independent of ϵ, which is the size of the gradient, and σ, which determines how narrow the valley is.

Asymptotically, as k grows large we have

$$\mathbf{c}_k \approx \left(\hat{c}_1 - \hat{b}_1 k\right) \mathbf{1} - \hat{b}_\lambda \frac{1}{1-\lambda} \mathbf{w}_\lambda \,.$$

For k large, each pass (3 micro-steps) will move approximately by:

$$|\Delta \mathbf{x}| = |\mathbf{x}^{k+1} - \mathbf{x}^k| \approx |\hat{b}_1| = \frac{\epsilon}{\sigma} \frac{1}{p_1 p_2 + p_2 p_3 + p_1 p_3} \,. \tag{22}$$

As with the $d = 2$ case in Section 2.3.2, the dependence on ϵ and σ is as ϵ/σ, which is small when $\sigma \gg \epsilon$. Since $p_1 + p_2 + p_3 = 1$, either zero, one, or two of the p_i can be small.

- If none of the p_i are small, then the scalar depending on the p_i cannot speed the progress. It is elementary to show that (22) is convex and is minimized by the symmetric case $p_1 = p_2 = p_3 = 1/3$, corresponding to $|\mathbf{v}_1| = |\mathbf{v}_2| = |\mathbf{v}_3|$. Thus, the slowest rate of progress along the valley is
$$\min |\Delta \mathbf{x}| = 3\epsilon/\sigma \,.$$
- If one of the p_i is small, then the BCD algorithm will nearly project onto the hyperplane spanned by the two non-zero coordinate blocks. The trajectory will then zig-zag near that plane with essentially the same rate as the $d = 2$ case in Section 2.3.2. In particular, if we set $p_3 = 0$ in (22), then it reduces to (20). If we then set $p_1 = p_2 = 1/2$, then the slowest rate of progress is $4\epsilon/\sigma$.
- If two of the p_i are small, then (22) can be large.

Numerically, in $\mathbb{R}^3$ we find that (22) is no more than twice the minimum value for 85.3% of unit vectors $\mathbf{v}$ and it is within the same order as the minimum for 96.2% of all unit vectors (using a spherically symmetric uniform distribution on the sphere $|v_1|^2 + |v_2|^2 + |v_3|^2 = 1$). Both of these percentages are significantly larger than the corresponding percentages for the case $n = d = 2$. Thus we observe that the portion of all directions that make BCD slow grows markedly as the number of blocks increases from 2 to 3.

The BCD method and the sequential recurrence (31) are not symmetric in the blocks of coordinates and so the choice of the order of optimization may matter. Progress along the valley direction is achieved by decreasing the coefficients $\{c_i\}$ and so optimizing first with respect to the largest c_i will result in the largest progress down the valley among all d possible first micro-steps. However, in the context of a real problem the coefficients $\{c_i\}$ will not be known and one might hope for a proxy, such as the value of the objective function f. In Section 2.3.2 we observed that for two coordinate blocks ($d = 2$), it is advantageous to use a greedy strategy (with respect to f)

to choose the order of micro-steps. For $d \geq 3$ this is no longer true as we will show in the following example.

Fix $d \geq 3$ and consider the initial state $\mathbf{x}^0 = (\mathbf{0}, \mathbf{v}_2, \mathbf{v}_3, \ldots, \mathbf{v}_d) \in W$, which can be described by the coefficients $(c_1, c_2, \ldots, c_d) = (0, 1, 1, \ldots, 1)$. If we optimize with respect to $\mathbf{x}_1$ first, then by direct calculation c_1 becomes

$$c_1 = 1 - \frac{d}{d-1}\frac{\epsilon}{\sigma}\,.$$

and the updated vector is $\mathbf{y}_1 = (c_1\mathbf{v}_1; \mathbf{v}_2; \ldots; \mathbf{v}_d)$. If instead we optimize with respect to $\mathbf{x}_2$ first, then c_2 becomes updated to

$$c_2 = 1 - \frac{1}{d-1} - \frac{d}{d-1}\frac{\epsilon}{\sigma}\,,$$

which produces a new vector $\mathbf{y}_2 = (\mathbf{0}; c_2\mathbf{v}_2; \mathbf{v}_3; \ldots; \mathbf{v}_d)$. Comparing the objective function $f(\mathbf{x})$ evaluated at the new points $\mathbf{y}_1$ and $\mathbf{y}_2$, we find that

$$\begin{aligned} f(\mathbf{y}_1) &= \epsilon - \frac{1}{d-1}\frac{\epsilon^2}{2\sigma} \quad \text{and} \\ f(\mathbf{y}_2) &= \epsilon\frac{d-2}{d-1} - \frac{1}{d-1}\frac{\epsilon^2}{2\sigma} + \frac{(d-2)\sigma}{2d(d-1)} = f(\mathbf{y}_1) + \frac{1}{d-1}\left(\frac{\sigma(d-2)}{2d} - \epsilon\right)\,. \end{aligned}$$

For $d \geq 3$ and $\sigma(d-2)/2d > \epsilon$ (in particular if $\epsilon \ll \sigma$ as in a narrow valley) then we will have $f(\mathbf{y}_1) < f(\mathbf{y}_2)$. Thus the greedy choice is to optimize with respect to $\mathbf{x}_1$ first. However,

$$\mathbf{v}^T\mathbf{y}_1 = 1 - \frac{1}{d-1}\frac{\epsilon}{2\sigma} \quad \text{while} \quad \mathbf{v}^T\mathbf{y}_2 = \frac{d-2}{d-1} - \frac{1}{d-1}\frac{\epsilon}{2\sigma} = \mathbf{v}^T\mathbf{y}_1 - \frac{2}{d-1}\,,$$

so optimizing with respect to $\mathbf{x}_2$ first gives more movement in the direction $-\mathbf{v}$, which is better for progress down the valley and allows for fewer steps to leave the valley. In fact, we may observe that $\mathbf{v}^T\mathbf{y}_1 > \mathbf{v}^T\mathbf{x}^0 = 1 - 1/d$ provided $\epsilon/\sigma < 2(d-1)/d$, i.e., $\mathbf{y}_1$ is further up the valley than the initial point.

This happens because of the composition of the objective function. The first part of f, with coefficient ϵ, measures linear distance in the direction $\mathbf{v}$. The second part, multiplied by σ, penalizes the square of the distance from V. If $\mathbf{x}$ is not close to V, then the objective f will be dominated by the quadratic part. A greedy strategy will tend to pull the iteration toward V, but not necessarily further down the valley in the direction $-\mathbf{v}$.

Summarizing, in the case $d = 3$ we find:

- Iterations are attracted linearly to a neighborhood of the valley floor with a rate of convergence per pass no worse than 1/8, independent of ϵ and σ.
- Iterations zig-zag at a distance $\sim \epsilon/\sigma$ from the valley floor.
- In one pass, iterations move a distance $\sim \epsilon/\sigma$ with a minimum of $3\epsilon/\sigma$.
- The slowest convergence to the valley and the slowest progress along the valley occur when $\mathbf{v}$ is maximally diagonal, i.e. $|\mathbf{v}_1| = |\mathbf{v}_2| = |\mathbf{v}_3|$.
- For $d = 3$ a large set of directions $\mathbf{v}$ make progress of alternating algorithms slow. The percentage of slow directions is higher than for the case $d = 2$.
- For $d > 2$, a greedy strategy based on decreasing the objective function may be counterproductive in a narrow valley.

2.3.4 Progress rate for $d \geq 2$

In previous sections we derived the progress rate $\hat{b}_1$ in a symmetric valley in the cases $d = 2$ (20) and $d = 3$ (22). Now we produce the progress rate for d arbitrary. We will use the results to show that the progress rate is slow except for a set of directions $\mathbf{v} \in \mathbb{R}^n$ that is exponentially small in n.

Theorem 2.3. *The progress rate is*

$$\hat{b}_1 = -\frac{\epsilon}{\sigma} \frac{1}{\sum_{i=1}^{d} \sum_{j=1}^{i-1} p_i p_j} = -\frac{\epsilon}{\sigma} \frac{2}{\sum_{i \neq j}^{d} p_i p_j} \, . \tag{23}$$

Proof. We give the high-level arguments of the proof now, while deferring calculations.

Corollary 2.2 showed that the eigenvalue 1 is simple and has eigenvector $\mathbf{1}$, and all other eigenvalues satisfy $|\lambda| < 1$. Thus, by the power method, $\hat{z}_1 \mathbf{1} = \lim_{k\to\infty} M^k \mathbf{z}$ for any $\mathbf{z}$, so in particular $\hat{b}_1 \mathbf{1} = \lim_{k\to\infty} M^k \mathbf{b}$. The matrix $M^\infty = \lim_{k\to\infty} M^k$ is thus the projector onto the component of a vector in the eigenvector $\mathbf{1}$ when expanded into the eigenvectors of M. We can thus write $M^\infty = \mathbf{1}\mathbf{u}^T$ for some $\mathbf{u}$.

Since $M^\infty M = M^\infty$, we know $\mathbf{1}\mathbf{u}^T M = \mathbf{1}\mathbf{u}^T$ and hence $\mathbf{u}^T M = \mathbf{u}^T$. Thus $\mathbf{u}$ is a left eigenvector of M with eigenvalue 1. Since this eigenvalue is simple, $\mathbf{u}$ is unique up to a scalar factor. In Lemma 2.4 we will show that the vector $\mathbf{y}$ with entries $y_i = p_i \sum_{j=1}^{i-1} p_j$ is a left eigenvector of M with eigenvalue 1. Since $M^\infty M^\infty = M^\infty$, we know $\mathbf{u}^T \mathbf{1} = 1$ and thus $\mathbf{u} = \mathbf{y}/(\mathbf{y}^T \mathbf{1})$.

In Lemma 2.5 we will show that $\mathbf{y}^T \mathbf{b} = -\epsilon/\sigma$. Thus $\hat{b}_1 = \mathbf{u}^T \mathbf{b} = -\epsilon/(\sigma \mathbf{y}^T \mathbf{1})$. We can compute directly that $\mathbf{y}^T \mathbf{1} = \sum_{i=1}^{d} p_i \sum_{j=1}^{i-1} p_j$, thus yielding (23). □

Lemma 2.4. *The vector*

$$\mathbf{y} = \sum_{i=1}^{d} p_i \left(\sum_{j=1}^{i-1} p_j \right) \mathbf{e}_i \tag{24}$$

is a left eigenvector of M in (18) with eigenvalue 1.

Proof. We will show that

$$\mathbf{y}^T M_d \cdots M_k = \mathbf{y}^T + \sum_{i=k}^{d} p_i (\mathbf{p} - \mathbf{e}_i)^T \tag{25}$$

for $k = 1, 2, \ldots, d$. Setting $k = 1$ then yields

$$\mathbf{y}^T M = \mathbf{y}^T + \sum_{i=1}^{d} p_i (\mathbf{p} - \mathbf{e}_i)^T = \mathbf{y}^T + \left(\sum_{i=1}^{d} p_i \right) \mathbf{p}^T - \sum_{i=1}^{d} p_i \mathbf{e}_i^T = \mathbf{y}^T \, ,$$

which means $\mathbf{y}$ is a left eigenvector with eigenvalue 1.

The argument is recursive, and so acts like a finite induction down in k. The base case is obtained by setting $k = d+1$ in (25), which yields $\mathbf{y}^T = \mathbf{y}^T$. The recursive step is then to take the $k+1$ version of (25), apply M_k on the right, and show we get (25). We thus compute

$$\left(\mathbf{y}^T + \sum_{i=k+1}^{d} p_i (\mathbf{p} - \mathbf{e}_i)^T \right) \left(I + \frac{1}{1 - p_k} \mathbf{e}_k \left(\mathbf{p} - \mathbf{e}_k \right)^T \right) \, .$$

The product of the first term with the identity I in the second term yields all of (25) except for its $i = k$ term. The remainder of the product is

$$\begin{aligned}
&\left(\mathbf{y}^T + \sum_{i=k+1}^{d} p_i(\mathbf{p} - \mathbf{e}_i)^T\right) \frac{1}{1-p_k}\mathbf{e}_k \left(\mathbf{p} - \mathbf{e}_k\right)^T \\
&= \left(p_k \left(\sum_{j=1}^{k-1} p_j\right) + \sum_{i=k+1}^{d} p_i p_k\right) \frac{1}{1-p_k} \left(\mathbf{p} - \mathbf{e}_k\right)^T \\
&= \left(\sum_{j=1,\neq k}^{d} p_j\right) \frac{p_k}{1-p_k} \left(\mathbf{p} - \mathbf{e}_k\right)^T = p_k \left(\mathbf{p} - \mathbf{e}_k\right)^T \ .
\end{aligned}$$

Since this is the desired $i = k$ term, the induction and the result are proven. □

Lemma 2.5. *For the vectors* $\mathbf{y}$ *in (24) and* $\mathbf{b}$ *in (19),* $\mathbf{y}^T\mathbf{b} = -\epsilon/\sigma$*.*

Proof. Substituting (25) into (19) yields

$$\begin{aligned}
\mathbf{y}^T\mathbf{b} &= \frac{-\epsilon}{\sigma} \sum_{i=1}^{d} \left(\mathbf{y}^T + \sum_{j=i+1}^{d} p_j(\mathbf{p} - \mathbf{e}_j)^T\right) \frac{1}{1-p_i}\mathbf{e}_i \\
&= \frac{-\epsilon}{\sigma} \sum_{i=1}^{d} \left(p_i \left(\sum_{j=1}^{i-1} p_j\right) + \sum_{j=i+1}^{d} p_j p_i\right) \frac{1}{1-p_i} \\
&= \frac{-\epsilon}{\sigma} \sum_{i=1}^{d} p_i \left(\sum_{j=1,\neq i}^{i-1} p_j\right) \frac{1}{1-p_i} = \frac{-\epsilon}{\sigma} \sum_{i=1}^{d} p_i = \frac{-\epsilon}{\sigma} \ .
\end{aligned}$$

□

From (23) we obtain the following.

Corollary 2.6. *The progress rate (23) is slowest when* $\mathbf{v}$ *is maximally diagonal, where (23) has value* $-(\epsilon/\sigma)2d/(d-1)$*.*

Proof. A direct computation shows that the function $\phi(\mathbf{p}) = \sum_{i\neq j}^{d} p_i p_j$ is concave down on the unit simplex $\sigma_d = \{\mathbf{p} : p_i \geq 0, \sum_{i=1}^{d} p_i = 1\}$. In particular, if we eliminate p_d from ϕ using the constraint and calculate second partial derivatives of $\bar{\phi}(p_1, \ldots, p_{d-1}) = \phi(\mathbf{p})$, we find that all mixed partial derivatives are -1, while the pure second derivatives all equal to -2. It is then easy to show that the Hessian matrix is strictly negative definite. Separately, the method of Lagrange multipliers shows that ϕ restricted to the unit simplex has a single critical point, which is the symmetric point $p_i = 1/d$ for all $1 \leq i \leq d$. The symmetric point, which corresponds to $\mathbf{v}$ being maximally diagonal, is the maximum point for ϕ and the minimum point for $\hat{b}_1$. □

In the previous section we saw that as n increases from 2 to 3, the fraction of directions that behave badly increases. Using Theorem 2.3 for $n = 3, 4, 5$ we calculated the values of $P2$ and $P10$, the portion of unit vectors for which the rate of progress $\hat{b}_1$ is greater than twice or 10 times the

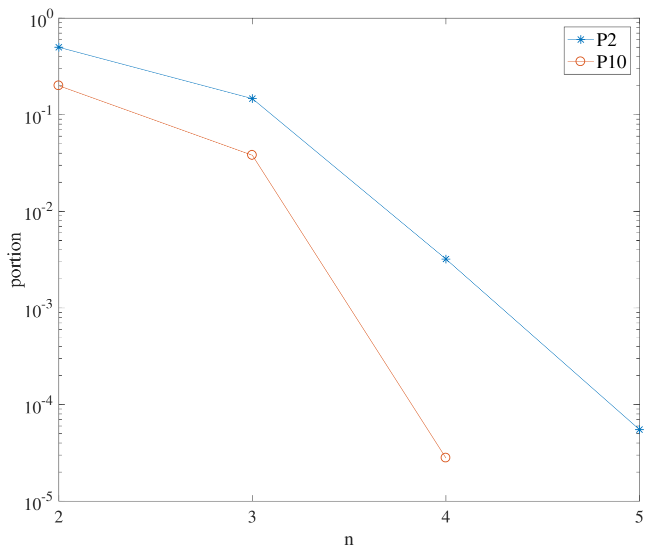


Figure 7: Log plot of the portion of vectors $\mathbf{v}$ on the unit sphere S^{n-1} for which the rate of progress, $\hat{b}_1$, is more than twice (P2 $*$) and 10 times (P10 $\circ$) the minimum value, $2n/(n-1)\frac{\epsilon}{\sigma}$, as a function of n for CD. For practical purposes, almost all directions $\mathbf{v}$ behave nearly as badly as a maximally diagonal direction.

slowest rate, respectively, and plotted the results in Figure 7. We see that even for $n = 4$ these portions are quite small. Thus, practically all directions are bad for a CD algorithm. The following results show that this pattern continues asymptotically for BCD. Proposition 2.7 shows that fast progress requires $\mathbf{v}$ to have a large projection onto a single block. Proposition 2.9 then shows that the set of such vectors is exponentially small in n.

Proposition 2.7. *For any $\alpha > 2$, the directions $\mathbf{v} \in S^{n-1}$ for which $|\hat{b}_1| > \alpha 2d/(d-1)\frac{\epsilon}{\sigma}$ satisfy*

$$p_i > \frac{1}{2} + \frac{1}{2}\sqrt{1 - \frac{2(d-1)}{\alpha d}} \qquad \textit{for some } i \textit{ with } 1 \leq i \leq d. \tag{26}$$

Proof. Without loss of generality, we can order $\{p_i\}$ so that $p_i \leq p_d$ for all $1 \leq i < d$. Note that for $\mathbf{p}$ satisfying $p_1 + \cdots + p_d = 1$, we have

$$\phi_d(\mathbf{p}) = \sum_{i \neq j}^{d} p_i p_j = \sum_{i \neq j}^{d-1} p_i p_j + p_d \sum_{i=1}^{d-1} p_i = \phi_{d-1}(\mathbf{p}') + p_d(1 - p_d)\,, \tag{27}$$

where we use $\mathbf{p}'$ to denote the vector formed from $\mathbf{p}$ by removing the last entry. We have that $\hat{b}_1 > \alpha 2d/(d-1)\frac{\epsilon}{\sigma}$ if and only if $\phi(\mathbf{p}) < (d-1)/2\alpha d$. From (27), this implies $p_d(1-p_d) < \frac{d-1}{2\alpha d}$. This inequality has solutions

$$p_d > \frac{1}{2} + \frac{1}{2}\sqrt{1 - \frac{2(d-1)}{\alpha d}} \quad \text{or} \quad p_d < \frac{1}{2} - \frac{1}{2}\sqrt{1 - \frac{2(d-1)}{\alpha d}} \tag{28}$$

provided $\alpha > 2(d-1)/d$. If we take $\alpha > 2$, then there are solutions for any d. The first solution is the desired inequality (26).

The second solution to (28) implies $p_d < 1/2$; since we assumed p_d was the largest p_i, this implies all p_i are less than $1/2$. Consider ϕ restricted to the unit simplex with the additional constraints $0 \le p_i \le 1/2$ for all i. Since ϕ is concave down on the simplex and since the constraints are linear, local minimum points must occur at the corners of the restricted region. The corners correspond to $k > 1$ coordinates equal to $1/k$ and the other $d-k$ coordinates equal to 0, i.e. permutations of the vector $\mathbf{q}_k$ that is $1/k$ in the first k coordinates and 0 elsewhere. We can compute directly

$$\phi(\mathbf{q}_k) = \frac{1}{k^2}\binom{i}{2} = \frac{k-1}{2k} = \frac{1}{2} - \frac{1}{2k} \ge \frac{1}{4}\,.$$

Since this is inconsistent with $\phi(\mathbf{p}) < (d-1)/2\alpha d < 1/4$, we can rule out the second solution to (28). □

Next we will apply Paul Lévy's *spherical isoparametric inequality* also known as *concentration of measure on the sphere* [25]. Let μ be the normalized surface area measure on S^{n-1} and let $\|\cdot\|$ denote Euclidean distance in $\mathbb{R}^n$.

Theorem 2.8 (Measure Concentration for the Sphere S^{n-1})**.** *Let $A \subset S^{n-1}$ be a measurable subset of the unit sphere S^{n-1} such that $\mu(A) = 1/2$. Let A_δ denote the δ-neighborhood of A in S^{n-1}. i.e., $A_\delta = \{x \in S^{n-1} \mid \exists z \in A, \|x - z\| \le \delta\}$. Then,*

$$\mu(A_\delta) \ge 1 - 2e^{-n\delta^2/2}\,. \tag{29}$$

Proposition 2.9. *If $\mathbf{v} \in S^{n-1}$ is partitioned into d blocks, the number of coordinates in each block is at most $n/2$, and $0 < \gamma < 1/2$, then the measure of the set of $\mathbf{v}$ for which $|\mathbf{v}_i|^2 \ge 1/2 + \gamma$ for some i is less than*

$$2d\exp(-n\gamma^2/2)\,. \tag{30}$$

Proof. We will show that the set $A = \{\mathbf{v} \in S^{n-1} \mid |\mathbf{v}_d|^2 \ge 1/2 + \gamma\}$ has measure less than $2\exp(-n\gamma^2/2)$. Permuting the distinguished index then yields (30). The set $B = \{\mathbf{v} \in S^{n-1} \mid |\mathbf{v}_d|^2 \le 1/2\}$ has measure at least 1/2 by the assumption that $\mathbf{v}_d$ contain not more than half the variables. By the isoparametric inequality (29) applied to B, $\mu(A) < 2\exp(-n\delta^2/2)$, where δ is the minimum distance between A and B. We thus need only show that $\delta \ge \gamma$.

The minimum distance between A and B occurs on their boundaries ∂A and ∂B. Defining

$$f(\mathbf{x}) = |\mathbf{x}_1|^2 + \cdots + |\mathbf{x}_{d-1}|^2 \quad \text{and} \quad g(\mathbf{x}) = |\mathbf{x}_d|^2\,,$$

we can characterize ∂A and ∂B as intersections of level sets of these functions as

$$\partial A = \{\mathbf{x} \mid f(\mathbf{x}) = 1/2 - \gamma,\ g(\mathbf{x}) = 1/2 + \gamma\} \quad \text{and} \quad \partial B = \{\mathbf{x} \mid f(\mathbf{x}) = 1/2,\ g(\mathbf{x}) = 1/2\}\,.$$

The minimal distance between ∂A and ∂B will occur at points $\mathbf{x} \in \partial A$ and $\mathbf{y} \in \partial B$ where the normal plane to ∂A at $\mathbf{x}$ intersects ∂B at $\mathbf{y}$. Note that the normal plane to ∂A at $\mathbf{x}$ is spanned by the vectors

$$\nabla f(\mathbf{x}) = (\mathbf{x}_1; \ldots; \mathbf{x}_{d-1}, \mathbf{0}) \quad \text{and} \quad \nabla g(\mathbf{x}) = (\mathbf{0}; \ldots; \mathbf{0}; \mathbf{x}_d).$$

From this condition we deduce that $\mathbf{y}$ must satisfy $\mathbf{y}_i = a\mathbf{x}_i$ for $1 \le i \le d-1$ and $\mathbf{y}_d = b\mathbf{x}_d$ for some real a and b. Applying the constraints $\mathbf{x} \in \partial A$ and $\mathbf{y} \in \partial B$, we obtain

$$a = \pm\frac{1}{\sqrt{1-2\gamma}} \quad \text{and} \quad b = \pm\frac{1}{\sqrt{1+2\gamma}}.$$

We can substitute these expressions directly into the distance formula to find that the distance is independent of the point $\mathbf{x} \in \partial A$ and is minimized at $\mathbf{y}$ corresponding to the positive values of a and b. For these points we find that

$$\delta^2 = d(\mathbf{x},\mathbf{y})^2 = 2 - \sqrt{1-2\gamma} - \sqrt{1+\gamma}.$$

Since $\gamma < 1/2$, we can expand the square root expressions as convergent Taylor series. The odd terms in the expansion cancel leaving

$$d(\mathbf{x},\mathbf{y})^2 = \gamma^2 + \frac{5}{4}\gamma^4 + \ldots,$$

where all the successive even powers have positive coefficients and we can conclude that $\delta > \gamma$. $\square$

2.3.5 The case $d > 3$ with $p_i = 1/d$

For $d > 3$ we will further analyze the maximally diagonal case $p_i = |\mathbf{v}_i|^2 = 1/d$ for all $1 \le i \le d$, which has the slowest progress along the valley. The update (14) in this case becomes

$$c_i = \frac{1}{d-1}\sum_{j=1,\neq i}^{d} c_j - \frac{d}{d-1}\frac{\epsilon}{\sigma}\,. \tag{31}$$

If we let s^j be the sequence $s^{dk+i} = c_i^k$, then s^j satisfies the recurrence

$$s^j = \frac{1}{d-1}\sum_{i=1}^{d-1} s^{j-i} - \frac{d}{d-1}\frac{\epsilon}{\sigma}\,. \tag{32}$$

This recurrence has a particular solution $s^j_{\text{part}} = -\frac{2}{d-1}\frac{\epsilon}{\sigma}j$, so that on a pass of BCD it moves in the valley direction by

$$|\Delta\mathbf{x}| = \frac{2d}{d-1}\frac{\epsilon}{\sigma}\,, \tag{33}$$

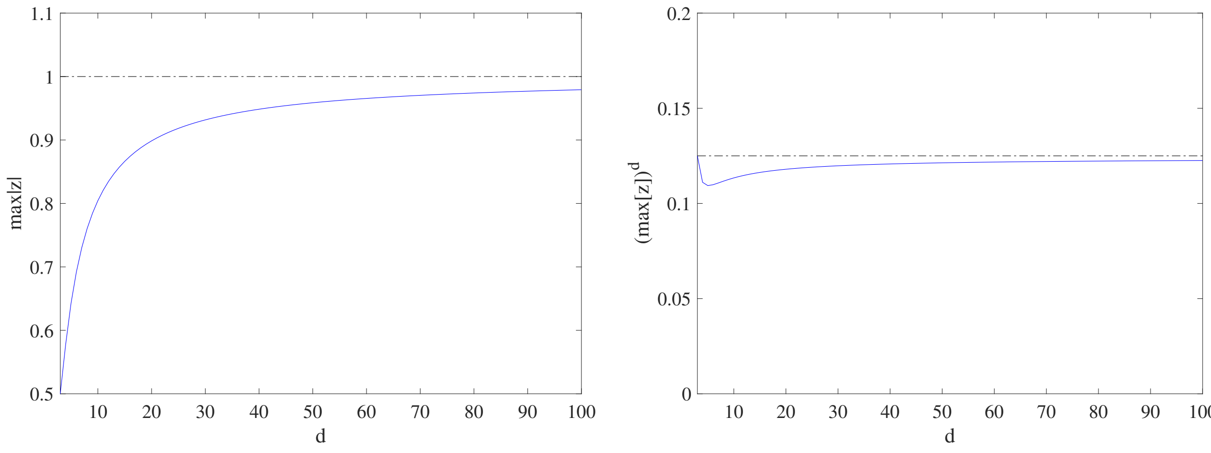
which agrees with (23) for $p_i = 1/d$. The homogeneous part of the recurrence (32) has the auxiliary equation

$$(d-1)z^{d-1} - z^{d-2} - z^{d-3} - \ldots - z - 1 = 0\,.$$

Dividing it by $z-1$ yields

$$(d-1)z^{d-2} + (d-2)z^{d-3} + \ldots + 2z + 1 = \sum_{i=0}^{d-2}(i+1)z^i = 0\,. \tag{34}$$

The recurrence (32) is equivalent to applying M_1, followed by a cyclic permutation of coordinates $c_i \mapsto c_{(i+1 \mod d)}$, which can be represented by a permutation matrix C. Now consider that



(a) (b)

Figure 8: (a) Maximal modulus of non-unity roots of the auxiliary polynomial (34) as a function of d, $d \geq 3$. For $d = 3$, $-1/2$ is the second root. As d becomes large the modulus of the root appears to approach 1, but relatively slowly. These roots represent the rate of contraction toward the diagonal (in coefficient space) of one micro-step of an alternating algorithm. (b) The modulus of the maximal root raised to the power d, $3 \leq d \leq 100$. This is an upper bound on the contraction rate to the valley of a pass of the algorithm. It appears this rate is bounded above by 1/8.

$M_i = C^{-i+1} M_1 C^{i-1}$. It is then easy to show, using $C^{-d+1} = C$, that the full matrix $M = M_d \cdots M_1$ is equal to $(CM_1)^d$. The polynomial in (34) is the characteristic polynomial of CM_1 divided by $z(z-1)$. The eigenvalues of M are thus the roots of (34) raised to the power d, along with 0 and 1. In Figure 8(a) we plot the maximum modulus of the roots as a function of d. In Figure 8(b) we plot the same maximal root raised to the power d for each d and observe that this is bounded above by 1/8 for $d = 3, \ldots, 100$. As discussed in Section 2.2, the maximum modulus of the eigenvalues of M other than 1 controls the rate of convergence to a neighborhood of the diagonal.

For the case $d > 3$ with $p_i = 1/d$, we find:

- Iterations are attracted to a neighborhood of the valley floor at a rate no worse than 1/8 per pass, independent of d, ϵ and σ.

We also confirmed that a pass moves the point a distance $\sim \epsilon/\sigma$ with a minimum of $\dfrac{2d}{d-1}\dfrac{\epsilon}{\sigma}$.

2.3.6 Comparison with Gradient Descent with Line Search

The GDLS algorithm computes the gradient and then moves along the ray in the negative gradient direction until it encounters the first local minimum. Since this algorithm is independent of the coordinate system, the progress rate of GDLS is independent of $\mathbf{v}$. In [13, Section 2.3.2], the behavior of GDLS in a valley in $d = 2$ was studied. By rotating the coordinate system to map $\mathbf{v}$ to $\mathbf{e}_1$ and $\mathbf{x}^0$ into the span of $\mathbf{e}_1$ and $\mathbf{e}_2$, the analysis also holds for a symmetric valley in $d > 2$. The iterates alternate between the two lines $x_2 = x_2^0$ and $x_2 = -(\epsilon/\sigma)^2/x_2^0$. The mean progress (along

$-\mathbf{e}_1$) is

$$\frac{\epsilon}{\sigma}\left(1+\frac{1}{2}\left(\left(\frac{\epsilon}{\sigma x_2^0}\right)^2+\left(\frac{\epsilon}{\sigma x_2^0}\right)^{-2}\right)\right) .$$

The progress rate depends on the initial value x_2^0 and attains its minimum value of $2\epsilon/\sigma$ when $|x_2^0| = \epsilon/\sigma$.

In comparison:

- Both methods move proportionally to ϵ/σ.
- The BCD progress rate is insensitive to the starting point but is sensitive to $\mathbf{v}$. Since the proportion of $\mathbf{v}$ yielding slow progress increases with d, one is likely to notice the effect of valleys when using BCD.
- The GDLS progress rate is insensitive to $\mathbf{v}$ but is sensitive to the starting point. GDLS does not have a mechanism for converging to the vicinity of the valley floor. Whether or not one notices the effect of a valley thus depends on either an (un)lucky choice of starting point or the behavior of $f(\mathbf{x})$ outside the local validity of the valley model.

2.4 Diagonal Valley with asymmetric walls

2.4.1 The general setup and the special case of coordinate descent

If the attraction to V is not radially symmetric about V, then

$$H^{\perp} = \sum_{m=1}^{n-1} \sigma_m \mathbf{u}_m \mathbf{u}_m^T$$

with $\{\mathbf{u}_m\}$ orthonormal, $\mathbf{u}_m^T\mathbf{v} = 0$ for all m, and $\sigma_m > 0$. In order to compute the update (3), we need to compute

$$\left(H_{ii}^{\perp}\right)^{-1} = \left(\sum_{m=1}^{n-1} \sigma_m \mathbf{u}_{mi}\mathbf{u}_{mi}^T\right)^{-1} ,$$

where $\mathbf{u}_{mi}$ is the i-th block out of $\mathbf{u}_m$.

Since this is too general to allow a useful formula, we will only consider block size one, i.e. the CD case. In this context $\mathbf{u}_{mi}$ is just a scalar u_{mi} and the inverse is just the reciprocal. Let

$$\hat{\sigma}_{ij} = \frac{\sum_{m=1}^{d-1} \sigma_m u_{mi} u_{mj}}{\sum_{m=1}^{d-1} u_{mi}u_{mj}} , \tag{35}$$

which is a weighted average of the σ_m, with some negative weights when $i \neq j$. Let $\mathbf{U}$ be the matrix whose first $d-1$ rows are $\mathbf{u}_1$ through $\mathbf{u}_{d-1}$ and whose last is $\mathbf{v}$. This matrix is orthogonal (as is its transpose), so $\sum_{m=1}^{d-1} u_{mi}u_{mj} + v_iv_j = \delta_{ij}$. We can then interpret

$$H_{ii}^{\perp} = \sum_{m=1}^{d-1} \sigma_m u_{mi}^2 = \frac{\sum_{m=1}^{d-1} \sigma_m u_{mi}^2}{\sum_{m=1}^{d-1} u_{mi}^2}(1 - v_iv_i) = \hat{\sigma}_{ii}(1-p_i)$$

as the familiar factor $1 - p_i$ times a weighted average of the σ_m. Similarly, for $i \neq j$ we have $H_{ij}^{\perp} = \hat{\sigma}_{ij}(-v_iv_j)$.

We can then write (3) as

$$x_i = \frac{1}{\hat{\sigma}_{ii}(1-p_i)} \left(\sum_{j=1,\neq i}^{d} \hat{\sigma}_{ij} v_i v_j x_j - \epsilon v_i \right) .$$

Letting $x_i = c_i v_i$, we can reduce to the coefficient equation

$$c_i = \frac{1}{\hat{\sigma}_{ii}(1-p_i)} \left(\sum_{j=1,\neq i}^{d} \hat{\sigma}_{ij} p_j c_j - \epsilon \right) = \frac{\sum_{j=1,\neq i}^{d} \frac{\hat{\sigma}_{ij}}{\hat{\sigma}_{ii}} p_j c_j}{1-p_i} - \frac{\epsilon}{\hat{\sigma}_{ii}(1-p_i)} .$$

This update is analogous to (15) in that it contains an averaging term and a drift term; it reduces to (15) when $\sigma_m = \sigma$ for all m.

We can put this into vector form, as we did in Section 2.3.1 for the symmetric valley. Corresponding to Equations (16) and (17), we have

$$\begin{aligned} \mathbf{c}^{\text{new}} &= M_i \mathbf{c} - \frac{\epsilon}{\hat{\sigma}_{ii}(1-p_i)} \mathbf{e}_i \quad \text{with} \\ M_i &= \sum_{j=1,\neq i}^{d} \left(\mathbf{e}_j + \frac{\hat{\sigma}_{ij} p_j}{(1-p_i)\hat{\sigma}_{ii}} \mathbf{e}_i \right) \mathbf{e}_j^T . \end{aligned} \tag{36}$$

Corresponding to (19) we have

$$\mathbf{b} = -\epsilon \sum_{i=1}^{d} M_d \cdots M_{i+1} \frac{1}{\hat{\sigma}_{ii}(1-p_i)} \mathbf{e}_i . \tag{37}$$

As in the general case in Section 2.2 and the symmetric case in Section 2.3, we are interested in finding $\hat{b}_1$ and the maximum of $\{|\lambda| : |\lambda| < 1\}$. As in Corollary 2.2, M has an eigenvalue 1 with eigenvector $\mathbf{1}$ and all its other eigenvalues satisfy $|\lambda| < 1$.

2.4.2 A particular example in $\mathbb{R}^3$

In $\mathbb{R}^3$ consider the diagonal direction $\mathbf{v} = (1,1,1)^T/\sqrt{3}$, two orthogonal vectors $\mathbf{u}_1 = (1,-1,0)^T/\sqrt{2}$ and $\mathbf{u}_2 = (-1,-1,2)^T/\sqrt{6}$, and two eigenvalues $\sigma_1, \sigma_2 > 0$. Let

$$\begin{aligned} f(\mathbf{x}) &= c + \epsilon \mathbf{v} \cdot \mathbf{x} + \frac{\sigma_1}{2}(\mathbf{u}_1 \cdot \mathbf{x})^2 + \frac{\sigma_2}{2}(\mathbf{u}_2 \cdot \mathbf{x})^2 \quad \text{so} \\ \nabla f(\mathbf{x}) &= \epsilon \mathbf{v} + \sigma_1 (\mathbf{u}_1 \cdot \mathbf{x}) \mathbf{u}_1 + \sigma_2 (\mathbf{u}_2 \cdot \mathbf{x}) \mathbf{u}_2 . \end{aligned}$$

Using the notation $(x,y,z) \in \mathbb{R}^3$, CD starting with the x direction produces the recurrence

$$\begin{aligned} x^{k+1} &= \frac{3\sigma_1 - \sigma_2}{3\sigma_1 + \sigma_2} y^k + \frac{2\sigma_2}{3\sigma_1 + \sigma_2} z^k - \frac{2\sqrt{3}}{3\sigma_1 + \sigma_2} \epsilon , \\ y^{k+1} &= \frac{3\sigma_1 - \sigma_2}{3\sigma_1 + \sigma_2} x^{k+1} + \frac{2\sigma_2}{3\sigma_1 + \sigma_2} z^k - \frac{2\sqrt{3}}{3\sigma_1 + \sigma_2} \epsilon , \quad \text{and} \\ z^{k+1} &= \frac{x^{k+1} + y^{k+1}}{2} - \frac{\sqrt{3}}{2\sigma_2} \epsilon. \end{aligned}$$

When $\sigma_1 = \sigma_2$, this reduces to the form in Section 2.3.3 with $\mathbf{v} = (1,1,1)^T/\sqrt{3}$.

This sequential recurrence can be rewritten as a simultaneous recurrence

$$\begin{aligned}
x^{k+1} &= \frac{3\sigma_1 - \sigma_2}{3\sigma_1 + \sigma_2} y^k + \frac{2\sigma_2}{3\sigma_1 + \sigma_2} z^k - \frac{2\sqrt{3}}{3\sigma_1 + \sigma_2}\epsilon\,, \\
y^{k+1} &= \left(\frac{3\sigma_1 - \sigma_2}{3\sigma_1 + \sigma_2}\right)^2 y^k + \frac{12\sigma_1\sigma_2}{(3\sigma_1 + \sigma_2)^2} z^k - \frac{12\sigma_1\sqrt{3}}{(3\sigma_1 + \sigma_2)^2}\epsilon\,, \quad \text{and} \\
z^{k+1} &= \frac{3\sigma_1(3\sigma_1 - \sigma_2)}{(3\sigma_1 + \sigma_2)^2} y^k + \frac{9\sigma_1\sigma_2 + \sigma_2^2}{(3\sigma_1 + \sigma_2)^2} z^k - \frac{9\sigma_1^2 + 24\sigma_1\sigma_2 + 3\sigma_2^2}{2\sigma_2(3\sigma_1 + \sigma_2)^2}\sqrt{3}\epsilon.
\end{aligned}$$

We can write this in matrix-vector form as $\mathbf{x}^{k+1} = M\mathbf{x}^k + \mathbf{a}$. (We could write this in terms of the coefficients using $\mathbf{c} = \mathbf{x}\sqrt{3}$ and $\mathbf{b} = \mathbf{a}\sqrt{3}$, but here we stay with $\mathbf{x}$.) The matrix M has eigenvalues 0 and 1 with eigenvectors $\mathbf{e}_1 = (1,0,0)^T$ and $\mathbf{v}$, respectively. For $\sigma_2 = 3\sigma_1$, 0 is a repeated eigenvalue with eigenvectors $\mathbf{e}_1$ and $\mathbf{e}_2$. Assuming $\sigma_2 \neq 3\sigma_1$, the third eigenvalue/eigenvector pair is

$$\lambda = \frac{-\sigma_2(3\sigma_1 - \sigma_2)}{(3\sigma_1 + \sigma_2)^2} \quad \text{and} \quad \mathbf{w}_\lambda = \left(\frac{3\sigma_1 + \sigma_2}{3\sigma_1 - \sigma_2}, -\frac{2\sigma_2}{3\sigma_1 - \sigma_2}, \frac{1}{2}\right)^T .$$

For the two cases $0 < \sigma_2 < 3\sigma_1$ and $\sigma_2 > 3\sigma_1$ it is elementary to check that $|\lambda| < 1$. Since $\lambda \to 1$ as $\sigma_1/\sigma_2 \to 0$, no better bound is possible; the case $\sigma_1/\sigma_2 \approx 0$ corresponds to f acting like an ill-conditioned sink in the direction orthogonal to $\mathbf{v}$.

Assume from hereon that $\sigma_2 \neq 3\sigma_1$. In the basis of eigenvectors we have $\mathbf{a} = \hat{a}_0\mathbf{e}_1 + \hat{a}_1\mathbf{v} + \hat{a}_\lambda\mathbf{w}_\lambda$, where

$$\hat{a}_0 = -\frac{\sqrt{3}}{\sigma_2}\epsilon\,, \quad \hat{a}_1 = -\frac{6}{\sigma_1 + \sigma_2}\epsilon\,, \quad \text{and} \quad \hat{a}_\lambda = \frac{(3\sigma_1^2 + \sigma_2^2)(3\sigma_1 - \sigma_2)\sqrt{3}}{2\sigma_2(\sigma_1 + \sigma_2)(3\sigma_1 + \sigma_2)^2}\epsilon\,.$$

Suppose $\mathbf{x}^0 \in \mathbf{R}^3$ is the initial guess and $\mathbf{x}^0 = \hat{x}_0\mathbf{e}_1 + \hat{x}_1\mathbf{v} + \hat{x}_\lambda\mathbf{w}_\lambda$. Then, for $k > 0$,

$$\mathbf{x}^k = \hat{x}_1\mathbf{1} + \hat{x}_\lambda\lambda^k\mathbf{w}_\lambda - \sum_{n=0}^{k-1}(\hat{a}_1\mathbf{v} + \hat{a}_\lambda\lambda^n\mathbf{w}_\lambda) = (\hat{x}_1 - \hat{a}_1 k)\,\mathbf{v} + \left(\hat{x}_\lambda\lambda^k - \hat{a}_\lambda\frac{1-\lambda^k}{1-\lambda}\right)\mathbf{w}_\lambda.$$

We see that $\mathbf{x}_k$ approaches a neighborhood of the line spanned by $\mathbf{v}$. Asymptotically, $\mathbf{x}_k$ moves on each pass of CD by $\Delta\mathbf{x} = \hat{a}_1\mathbf{v}$. Thus, on each pass it moves a distance $|\Delta\mathbf{x}| = |\hat{a}_1||\mathbf{v}| = |\hat{a}_1| = 6\epsilon/(\sigma_1 + \sigma_2)$. For $\sigma_1 = \sigma_2$, this result reduces to the symmetric case. If σ_1 or σ_2 is much larger than the other, then the speed of the algorithm becomes approximately inversely proportional to the *largest* eigenvalue.

Note that $\sigma_1 + \sigma_2$ is the trace of the Hessian matrix. We will show in the next section that if $\mathbf{v}$ is maximally diagonal, then the progress is inversely proportional to the trace. We found that when $\mathbf{v} \neq (1,1,1)^T/\sqrt{3}$ the progress is inversely proportional to a weighted sum of the eigenvalues (not the trace), but the formulas are too cumbersome to be useful.

2.4.3 Progress in the maximally diagonal case

In the maximally diagonal case $v_i = 1/\sqrt{d}$ for all i, (35) becomes

$$\hat{\sigma}_{ij} = \frac{\sum_{m=1}^{d-1}\sigma_m u_{mi} u_{mj}}{\delta_{ij} - 1/d}\,. \tag{38}$$

Since the denominator is now constant, we can use properties of $\mathbf{u}_m$ to obtain properties that can be used to simplify later formulas. Since $\mathbf{u}_m$ is orthogonal to $\mathbf{v}$, which is constant, we have

$$\sum_{j=1}^{d}\sum_{m=1}^{d-1}\sigma_m u_{mi}u_{mj} = \sum_{m=1}^{d-1}\sigma_m u_{mi}\left(\sum_{j=1}^{d}u_{mj}\right) = \sum_{m=1}^{d-1}\sigma_m u_{mi}0 = 0\,. \tag{39}$$

We then have

$$\sum_{j=1}^{d}\hat{\sigma}_{ij} = \frac{1}{-1/d}\sum_{j=1}^{d}\sum_{m=1}^{d-1}\sigma_m u_{mi}u_{mj} + \left(-\frac{1-1/d}{-1/d}+1\right)\hat{\sigma}_{ii} = d\hat{\sigma}_{ii}$$

$$\text{so}\quad 1 = \frac{1}{(d-1)\hat{\sigma}_{ii}}\sum_{j=1,\neq i}^{d}\hat{\sigma}_{ij}\,. \tag{40}$$

The matrix M_i in (36) and vector $\mathbf{b}$ in (37) become

$$M_i = \sum_{j=1,\neq i}^{d}\left(\mathbf{e}_j + \frac{\hat{\sigma}_{ij}}{(d-1)\hat{\sigma}_{ii}}\mathbf{e}_i\right)\mathbf{e}_j^T \quad\text{and} \tag{41}$$

$$\mathbf{b} = \frac{-\epsilon d}{d-1}\sum_{i=1}^{d} M_d\cdots M_{i+1}\frac{1}{\hat{\sigma}_{ii}}\mathbf{e}_i\,. \tag{42}$$

Similar to the symmetric valley in Section 2.3.4, we can now determine the progress rate.

Theorem 2.10. *For an asymmetric, maximally diagonal valley, the progress rate is*

$$\hat{b}_1 = -\frac{\epsilon 2d}{\sum_{m=1}^{d-1}\sigma_m} = -\frac{\epsilon 2d}{\operatorname{trace}(H^{\perp})}\,.$$

Proof. The proof structure is the same as for Theorem 2.3. Lemma 2.11 shows that a specific $\mathbf{y}$ is a left eigenvector of M with eigenvalue one. Lemma 2.12 shows $\mathbf{y}^T\mathbf{b} = -\epsilon d^2$ and Lemma 2.13 shows $\mathbf{y}^T\mathbf{1} = (d/2)\sum_{m=1}^{d-1}\sigma_m = (d/2)\operatorname{trace}(H^{\perp})$. Computing $\hat{b}_1 = \mathbf{y}^T\mathbf{b}/\mathbf{y}^T\mathbf{1}$ yields the desired formula. $\square$

Lemma 2.11. *The vector*

$$\mathbf{y} = \sum_{i=1}^{d}\left(\sum_{j=1}^{i-1}\hat{\sigma}_{ij}\right)\mathbf{e}_i \tag{43}$$

is a left eigenvector of $M = M_d\cdots M_1$ (with M_i in (41)) with eigenvalue 1.

Proof. We will show that

$$\mathbf{y}^T M_d\cdots M_k = \sum_{j=1}^{k-1}\left(\sum_{m=1}^{j-1}\hat{\sigma}_{jm} + \sum_{m=k}^{d}\hat{\sigma}_{mj}\right)\mathbf{e}_j^T + \sum_{j=k+1}^{d}\left(\sum_{m=k}^{j-1}\hat{\sigma}_{mj}\right)\mathbf{e}_j^T \tag{44}$$

for $k = 1, 2, \ldots, d$. Setting $k = 1$ then yields

$$\mathbf{y}^T M = \mathbf{y}^T M_d \cdots M_1 = 0 + \sum_{j=2}^{d} \left(\sum_{m=1}^{j-1} \hat{\sigma}_{mj} \right) \mathbf{e}_j^T = \mathbf{y}^T ,$$

which means $\mathbf{y}$ is a left eigenvector with eigenvalue 1.

The argument is recursive, and so acts like a finite induction down in k. The base case is obtained by setting $k = d+1$ in (44), which yields $\mathbf{y}^T = \mathbf{y}^T$. The recursive step is then to take the $k+1$ case of (44), apply M_k on the right, and show we get (44). We thus compute

$$\left(\sum_{j=1}^{k} \left(\sum_{m=1}^{j-1} \hat{\sigma}_{jm} + \sum_{m=k+1}^{d} \hat{\sigma}_{mj} \right) \mathbf{e}_j^T + \sum_{j=k+2}^{d} \left(\sum_{m=k+1}^{j-1} \hat{\sigma}_{mj} \right) \mathbf{e}_j^T \right) \left(\sum_{i=1,\neq k}^{d} \left(\mathbf{e}_i + \frac{\hat{\sigma}_{ki}}{(d-1)\hat{\sigma}_{kk}} \mathbf{e}_k \right) \mathbf{e}_i^T \right) .$$

For the terms with $1 \le j \le k-1$ and $k+2 \le j \le d$, M_k acts as the identity. For the $j = k$ term we have

$$\begin{aligned} &\left(\sum_{m=1}^{k-1} \hat{\sigma}_{km} + \sum_{m=k+1}^{d} \hat{\sigma}_{mk} \right) \mathbf{e}_k^T \sum_{i=1,\neq k}^{d} \left(\mathbf{e}_i + \frac{\hat{\sigma}_{ki}}{(d-1)\hat{\sigma}_{kk}} \mathbf{e}_k \right) \mathbf{e}_i^T \\ &= \left(\sum_{m=1}^{k-1} \hat{\sigma}_{km} + \sum_{m=k+1}^{d} \hat{\sigma}_{mk} \right) \frac{1}{(d-1)\hat{\sigma}_{kk}} \sum_{i=1,\neq k}^{d} \hat{\sigma}_{ki} \mathbf{e}_i^T . \end{aligned}$$

Noting the symmetry $\hat{\sigma}_{mk} = \hat{\sigma}_{km}$ and using (40), the factor before the sum becomes one and we reduce this $j = k$ term to

$$\sum_{i=1,\neq k}^{d} \hat{\sigma}_{ki} \mathbf{e}_i^T = \sum_{j=1}^{k-1} \hat{\sigma}_{kj} \mathbf{e}_j^T + \sum_{j=k+1}^{d} \hat{\sigma}_{kj} \mathbf{e}_j^T .$$

Combining with the original $1 \le j \le k-1$ and $k+2 \le j \le d$ terms, we then have

$$\begin{aligned} &\sum_{j=1}^{k-1} \hat{\sigma}_{kj} \mathbf{e}_j^T + \sum_{j=k+1}^{d} \hat{\sigma}_{kj} \mathbf{e}_j^T + \sum_{j=1}^{k-1} \left(\sum_{m=1}^{j-1} \hat{\sigma}_{jm} + \sum_{m=k+1}^{d} \hat{\sigma}_{mj} \right) \mathbf{e}_j^T + \sum_{j=k+2}^{d} \left(\sum_{m=k+1}^{j-1} \hat{\sigma}_{mj} \right) \mathbf{e}_j^T \\ &= \sum_{j=1}^{k-1} \left(\hat{\sigma}_{kj} + \sum_{m=1}^{j-1} \hat{\sigma}_{jm} + \sum_{m=k+1}^{d} \hat{\sigma}_{mj} \right) \mathbf{e}_j^T + \hat{\sigma}_{k,k+1} \mathbf{e}_{k+1}^T + \sum_{j=k+2}^{d} \left(\hat{\sigma}_{kj} + \sum_{m=k+1}^{j-1} \hat{\sigma}_{mj} \right) \mathbf{e}_j^T . \end{aligned}$$

Combining terms yields (44), as desired. □

Lemma 2.12. *For* $\mathbf{y}$ *in (43) and* $\mathbf{b}$ *in (42),* $\mathbf{y}^T \mathbf{b} = -\epsilon d^2$.

Proof. Substituting (44) into (42) yields

$$\begin{aligned} \mathbf{y}^T \mathbf{b} &= \frac{-\epsilon d}{d-1} \sum_{i=1}^{d} \left(\sum_{j=1}^{i} \left(\sum_{m=1}^{j-1} \hat{\sigma}_{jm} + \sum_{m=i+1}^{d} \hat{\sigma}_{mj} \right) \mathbf{e}_j^T + \sum_{j=i+2}^{d} \left(\sum_{m=i+1}^{j-1} \hat{\sigma}_{mj} \right) \mathbf{e}_j^T \right) \frac{1}{\hat{\sigma}_{ii}} \mathbf{e}_i \\ &= \frac{-\epsilon d}{d-1} \sum_{i=1}^{d} \left(\sum_{m=1}^{i-1} \hat{\sigma}_{im} + \sum_{m=i+1}^{d} \hat{\sigma}_{mi} \right) \frac{1}{\hat{\sigma}_{ii}} = -\epsilon d \sum_{i=1}^{d} \frac{1}{\hat{\sigma}_{ii}(d-1)} \sum_{m=1,\neq i}^{d} \hat{\sigma}_{im} . \end{aligned}$$

Applying (40) then yields $-\epsilon d^2$ as desired. □

Lemma 2.13. *For* $\mathbf{y}$ *in (43),* $\mathbf{y}^T\mathbf{1} = (d/2)\sum_{m=1}^{d-1}\sigma_m = (d/2)\mathrm{trace}(H^\perp)$*.*

Proof. From the definitions of $\mathbf{y}$ in (43) and $\hat{\sigma}_{ij}$ in (38),

$$\begin{aligned}\mathbf{y}^T\mathbf{1} &= \sum_{i=1}^{d}\left(\sum_{j=1}^{i-1}\hat{\sigma}_{ij}\right) = \sum_{i=1}^{d}\left(\sum_{j=1}^{i-1}\frac{\sum_{m=1}^{d-1}\sigma_m u_{mi}u_{mj}}{\delta_{ij}-1/d}\right)\\ &= -d\sum_{i=1}^{d}\sum_{j=1}^{i-1}\sum_{m=1}^{d-1}\sigma_m u_{mi}u_{mj} = -\frac{d}{2}\sum_{i=1}^{d}\sum_{j=1}^{d}\sum_{m=1}^{d-1}\sigma_m u_{mi}u_{mj} + \frac{d}{2}\sum_{i=1}^{d}\sum_{m=1}^{d-1}\sigma_m u_{mi}^2\,.\end{aligned}$$

The first term is zero by (39). By the normalization convention $\sum_{i=1}^{d}u_{mi}^2 = 1$, the second term is $(d/2)\sum_{m=1}^{d-1}\sigma_m$, as desired. □

3 BCD Dynamics Near a Symmetric-sided Valley-like Sink or Saddle

In this section we consider how the analysis in symmetric-sided valleys in Section 2.3 relates to the sink and saddle cases. Recall from (15) that the valley update can be written

$$c_i = \frac{1}{1-p_i}\sum_{j=1,\neq i}^{d}c_jp_j - \frac{\epsilon}{\sigma(1-p_i)}\,, \tag{45}$$

where the first term acts to move the iteration toward the diagonal and the second term determines the progress down the valley. For concreteness we will compare with the special case $d = 2$ and $p_1 = p_2 = 1/2$, for which (45) yields

$$c_1 = c_2 - 2\frac{\epsilon}{\sigma} \quad \text{and} \quad c_2 = c_1 - 2\frac{\epsilon}{\sigma}\,.$$

In Section 3.1 we consider a hyperbolic sink or saddle, in Section 3.2 we consider a nonhyperpolic sink or saddle, and in Section 3.3 we consider an essential discontinuity that is sink-like or saddle-like. In the hyperbolic case an explicit update rule corresponding to (45) is easy to find, in the nonhyperbolic case it is possible to find but too ugly to be useful, and in the essential discontinuity case it is (seemingly) impossible to find. Instead we will derive implicit, approximate update rules in which c_i appears on both side. These allow better interpretation and qualitative matching to (45). The parameter ϵ, which measured the gradient along $\mathbf{v}$ in a valley, is now used to measure the strength of the attraction to or repulsion from $\mathbf{0}$ along $\mathbf{v}$. In all cases we find that the progress is proportional to $(\epsilon/\sigma)/(1-p_i)$, as it was for the valley. In the hyperbolic case we find that progress is proportional to the current distance to $\mathbf{0}$ and in the nonhyperbolic and essential discontinuity cases we find that it is proportional to the cube of this distance, and thus much slower.

3.1 Hyperbolic Sink or Saddle

Instead of the valley objective function (1), we consider

$$f(\mathbf{x}) = \epsilon\frac{(\mathbf{v}^T\mathbf{x})^2}{2} + \frac{1}{2}\mathbf{x}^TH^\perp\mathbf{x} = \epsilon\frac{(\mathbf{v}^T\mathbf{x})^2}{2} + \frac{\sigma}{2}(\mathbf{x}^T\mathbf{x} - (\mathbf{v}^T\mathbf{x})^2)\,,$$

which has a sink at $\mathbf{x} = \mathbf{0}$ if $\epsilon > 0$ and a saddle if $\epsilon < 0$. We can then compute $\nabla_{\mathbf{x}_i} f(\mathbf{x}) = \epsilon \mathbf{v}^T \mathbf{x} \mathbf{v}_i + \sigma(\mathbf{x}_i - \mathbf{v}^T \mathbf{x} \mathbf{v}_i)$, set $\nabla_{\mathbf{x}_i} f(\mathbf{x}) = \mathbf{0}$, and obtain a scalar coefficient equation

$$0 = \epsilon \sum_{j=1}^{d} c_j p_j + \sigma \left(c_i - \sum_{j=1}^{d} c_j p_j \right) .$$

Although one can solve for c_i to obtain an update rule, it is more enlightening to rearrange as

$$c_i = \frac{1}{1-p_i} \sum_{j=1,\neq i}^{d} c_j p_j - \frac{\epsilon}{\sigma(1-p_i)} \sum_{j=1}^{d} c_j p_j \tag{46}$$

and interpret it as an implicit, approximate update rule, where the right side uses the current value of c_i and the left side is the new value. As in (45), the first term acts to move the iteration toward the diagonal and the second term determines the progress toward the sink or away from the saddle, which are at $\mathbf{0}$. Progress is proportional to $(\epsilon/\sigma)/(1-p_i)$ and, when all $c_j \approx c$, to the distance of $\mathbf{c}$ to $\mathbf{0}$.

For $d = 2$ and $p_1 = p_2 = 1/2$, (46) yields

$$c_1 = c_2 - \frac{\epsilon}{\sigma}(c_1 + c_2) \quad \text{and} \quad c_2 = c_1 - \frac{\epsilon}{\sigma}(c_1 + c_2) .$$

The BCD algorithm will alternate between these two lines, going inward when $\epsilon > 0$ and outward when $\epsilon < 0$.

3.2 Nonhyperbolic Sink or Saddle

We now consider

$$f(\mathbf{x}) = \epsilon \frac{(\mathbf{v}^T\mathbf{x})^4}{4} + \frac{1}{2}\mathbf{x}^T H^{\perp} \mathbf{x} = \epsilon \frac{(\mathbf{v}^T\mathbf{x})^4}{4} + \frac{\sigma}{2}(\mathbf{x}^T\mathbf{x} - (\mathbf{v}^T\mathbf{x})^2) ,$$

which has a nonhyperbolic sink at $\mathbf{x} = \mathbf{0}$ if $\epsilon > 0$ and a nonhyperbolic saddle if $\epsilon < 0$. Following the same process of setting $\nabla_{\mathbf{x}_i} f(\mathbf{x}) = \mathbf{0}$ leads to the scalar coefficient equation

$$0 = \epsilon \left(\sum_{j=1}^{d} c_j p_j \right)^3 - \sigma \sum_{j=1}^{d} c_j p_j + \sigma c_i . \tag{47}$$

As a cubic, this equation is solvable for c_i, but the expression is ugly and not useful. Rearranging (47), we can obtain the implicit update rule

$$c_i = \frac{1}{1-p_i} \sum_{j=1,\neq i}^{d} c_j p_j - \frac{\epsilon}{\sigma(1-p_i)} \left(\sum_{j=1}^{d} c_j p_j \right)^3 . \tag{48}$$

As in (46), the first term acts to move the iteration toward the diagonal and the second term determines the progress toward the sink or away from the saddle, which are at $\mathbf{0}$. Progress is proportional to $(\epsilon/\sigma)/(1-p_i)$ and, when all $c_j \approx c$, to the cube of the distance of $\mathbf{c}$ to $\mathbf{0}$. Hence progress is much slower than in the hyperbolic case.

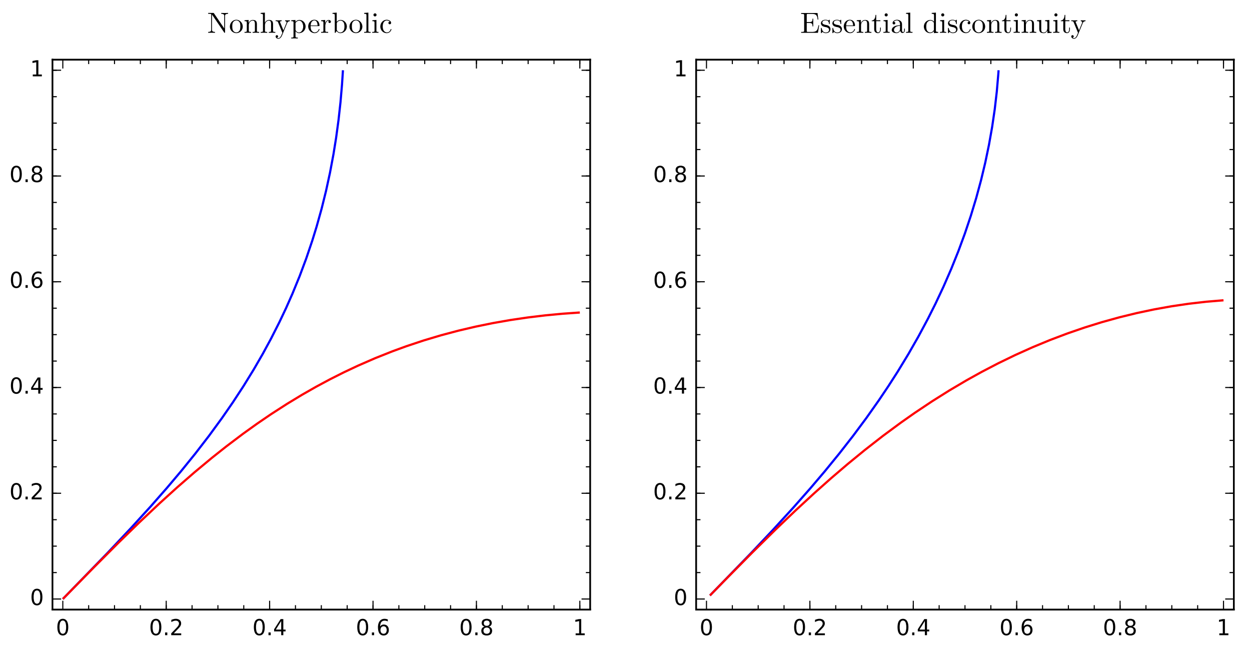


Figure 9: The curves defining the CD iteration for a sink or saddle in the nonhyperbolic and essential discontinuity cases. The left side shows the curves (49) with $|\epsilon/\sigma| = 1$ and the right side shows the curves (51) with $|\epsilon/\sigma| = 1/4$; although quite similar, the curves are not identical. For $\epsilon > 0$, $(0, 0)$ is a sink and iterations will slowly zig-zag into the cusp, with steps proportional to the cube of the distance to $(0, 0)$. For $\epsilon < 0$, $(0, 0)$ is a saddle and iterations will slowly zig-zag out with steps of the same size.

For $d = 2$ and $p_1 = p_2 = 1/2$, (48) yields

$$c_1 = c_2 - \frac{\epsilon}{\sigma}\left(\frac{c_1 + c_2}{2}\right)^3 \quad \text{and} \quad c_2 = c_1 - \frac{\epsilon}{\sigma}\left(\frac{c_1 + c_2}{2}\right)^3 . \tag{49}$$

The BCD algorithm will alternate between these two curves, which are plotted in Figure 9. The iterations will slowly zig-zag into ($\epsilon > 0$) or out of ($\epsilon < 0$) the cusp.

3.3 Sink or Saddle at an Essential Discontinuity

In [28], the extremely slow transient dynamics sometimes observed when BCD algorithms are used for tensor approximations is attributed to a particular peculiar feature in the objective function f. This feature is saddle-like in that it is attracting from most directions and repelling in one direction. The attraction phase is fast but the repulsion phase is very slow due to being in an extremely narrow valley. The center of this feature, which for convenience we will set as $\mathbf{0}$, is a point of discontinuity of f. The limit $\lim_{\mathbf{x}\to\mathbf{0}} f(\mathbf{x})$ does not exist, but the limit along any ray (i.e. $\lim_{t\to 0^+} f(t\mathbf{x})$) exists and is uniformly bounded. Thus f has a discontinuity at $\mathbf{0}$ that is not removable, so we call it essential.

For the particular model problem studied, it is shown [28, Lemma A.2] that

$$\lim_{t\to 0^+} f(t\mathbf{x}) = a_1 - a_2\left(\mathbf{v}^T\frac{\mathbf{x}}{|\mathbf{x}|}\right)^2 \quad \text{with } a_1 > 0 \text{ and } a_2 > 0\,.$$

Rearranging yields

$$(a_1 - a_2) + \frac{a_2}{|\mathbf{x}|^2}\left(\mathbf{x}^T\mathbf{x} - (\mathbf{v}^T\mathbf{x})^2\right)\,,$$

which shows how the distance from V appears. In principle the behavior along $\mathbf{v}$ could be of several forms, including non-differentiable behavior such as $\epsilon|\mathbf{v}^T\mathbf{x}|$. The example in [28] has hyperbolic behavior like $\epsilon(\mathbf{v}^T\mathbf{x})^2/2$, so we will use the normal form

$$f(\mathbf{x}) = \epsilon\frac{(\mathbf{v}^T\mathbf{x})^2}{2} + \frac{\sigma}{2|\mathbf{x}|^2}\left(\mathbf{x}^T\mathbf{x} - (\mathbf{v}^T\mathbf{x})^2\right)\,.$$

The constant σ in (12) is replaced by $\sigma/|\mathbf{x}|^2$, which goes to infinity as $\mathbf{x}\to\mathbf{0}$, so the valley is extremely narrow for small $\mathbf{x}$. We can then compute

$$\nabla f(\mathbf{x}) = \epsilon\mathbf{v}^T\mathbf{x}\mathbf{v} - \frac{\sigma\mathbf{v}^T\mathbf{x}}{(\mathbf{x}^T\mathbf{x})^2}(\mathbf{x}^T\mathbf{x}\mathbf{v} - \mathbf{v}^T\mathbf{x}\mathbf{x})\,.$$

From the (partial) gradient, we again see that setting $\nabla_{\mathbf{x}_i} f(\mathbf{x}) = \mathbf{0}$ leads to a scalar coefficient equation, which is

$$0 = \epsilon - \frac{\sigma}{\left(\sum_{j=1}^d p_j c_j^2\right)^2}\left(\sum_{j=1,\neq i}^d p_j c_j^2 - c_i\sum_{j=1,\neq i}^d p_j c_j\right)\,.$$

Rearranging to solve for the free c_i in the second term, we obtain the implicit update rule

$$c_i = \frac{\sum_{j=1,\neq i}^d p_j c_j^2}{\sum_{j=1,\neq i}^d p_j c_j} - \frac{\epsilon}{\sigma}\frac{\left(\sum_{j=1}^d p_j c_j^2\right)^2}{\sum_{j=1,\neq i}^d p_j c_j}\,. \tag{50}$$

If all $c_j \approx c$, then $\sum_{j=1,\neq i}^d p_j c_j \approx (1-p_i)c$, $\sum_{j=1,\neq i}^d p_j c_j^2 \approx (1-p_i)c^2$, and $\sum_{j=1}^d p_j c_j^2 \approx c^2$, so (50) acts as $c_i \approx c - (\epsilon/\sigma)/(1-p_i)c^3$. The first term is still a weighted average of the c_j (using $p_j c_j$ as the weight on c_j) and so acts to move the iteration toward the diagonal. The second term determines the progress toward the sink or away from the saddle, with progress proportional to $(\epsilon/\sigma)/(1-p_i)$ and to the cube of the distance of $\mathbf{c}$ to $\mathbf{0}$. Hence progress is similar to the nonhyperbolic case.

For $d = 2$ and $p_1 = p_2 = 1/2$, (50) yields

$$c_1 = c_2 - \frac{\epsilon}{\sigma}\frac{(c_1^2 + c_2^2)^2}{c_2} \quad \text{and} \quad c_2 = c_1 - \frac{\epsilon}{\sigma}\frac{(c_1^2 + c_2^2)^2}{c_1}\,. \tag{51}$$

We plot these curves in Figure 9 and note their similarity to the nonhyperbolic case.

4 Discussion and Implications

4.1 Primary Conclusions

One can think of iterations of BCD in a valley as the cross product of slow descent in one direction $-\mathbf{v}$ and contraction to a sink in the directions orthogonal to $\mathbf{v}$. While iterations are not too close to the bottom of the valley, they decompose into these two motions. However, near the bottom of the valley these two motions interact and iterations zig-zag near the valley floor. Our analysis reveals the following insights into optimization in a valley:

- When the sink is well-conditioned, convergence to a vicinity of the valley floor is linear, with small rate of convergence.
- When the sink is poorly-conditioned, convergence to a vicinity of the valley floor is still linear, but the rate can be close to 1. This situation is to be expected, since convergence to a minimum is slower for a poorly-conditioned sink, even without the effects of a valley.
- Once the iteration is close to the valley floor, it zig-zags at distance from the valley floor that is proportional to strength of descent and inversely proportional to the strength of the sink.
- The progress of BCD (as well as GDLS) is proportional to the strength of the descent and inversely proportional to the strength of the sink.
- The progress of BCD for most valley directions $\mathbf{v}$ is nearly as bad as the worst-case (maximally diagonal) direction. Thus we find that valleys are a phenomenon that might frequently be encountered in high dimensional problems such as tensor approximation.
- When starting the algorithm, the ordering of blocks in the first few steps can be important. A greedy approach that chooses the micro-step that gives the best improvement in the objective function may in fact give the worst progress along the downhill direction of the valley and thus lead to slower transient through the valley.
- The behavior near sinks and saddles is consistent with the behavior in a valley. Progress in the nonhyperbolic and essential discontinuity cases are similar to each other and much slower than in the hyperbolic case.

4.2 Algorithmic Implications

One immediate implication is that greedy methods (e.g. [6]) may not always work well. For $d > 2$, a greedy first step can push the iteration further up the valley, instead of down.

A second implication is that simply continuing with BCD when a valley is encountered will lead to poor performance. Instead the algorithm should be designed to detect when it has entered a narrow valley and then take some kind of evasive action.

Certain existing ideas, independent of the BCD approach, are useful to improve progress in valleys.

- When in a valley, local optimization tends to produce updates that move in the correct (down-valley) direction but not very far. By extrapolating (over-relaxing) to take bigger steps, the overall progress rate is increased.

- After some number of iterations have been performed, the general direction of the valley can be established by fitting to the trend in the iterations. Then one can extrapolate in that direction or perform a line search (or higher-order search) in that direction. We think that the benefit of naive extrapolation is limited because the problem is "doubly ill-conditioned":
  - determining the downhill direction in a narrow valley is ill-conditioned in that small changes in the position produce large changes in the gradient direction and
  - given a presumptive down-valley direction, a line search in that direction is also ill-conditioned.

  One may not succeed in moving far down the valley because the line direction, unless it is extremely accurate, will quickly lead up the side of the valley and far steps are rejected since they increase the objective function.

- A method intended to overcome the specific difficulties of extrapolation in narrow valleys was proposed in [21]. Their method is to search by attempting a long parabolic extrapolation, but rather than rejecting/accepting the step from the objective function at the end of the extrapolation, follow the extrapolation by a normal, un-extrapolated step, and then test the objective function. This added relaxation step, according to their reasoning, will move a point that has veered slightly up the side back down into the valley again.

- By extending the standard proof that conjugate gradient (CG) (see e.g. [35, 11, 12]) solves a $n \times n$ linear system exactly in n steps, one can show that nonlinear conjugate gradient (NCG) will find the bottom of a straight valley in a finite number of steps and then move infinitely far down such an idealized valley[1]. NCG methods for the tensor approximation problem have already been developed (e.g. [33, 1, 10, 44]).

- As noted in [28, Section 5.2] (and anticipated by [38, 32, 31]) adding regularization to the error function in the tensor approximation problem removes the essential discontinuity and the narrowest portion of the valley.

When BCD is being considered for some application, there is usually some specific reason for it, such as the ease of updating one block in ALS, that discourages one from simply abandoning BCD. This motivates the development of hybrid methods. For the tensor approximation problem, algorithms to have already been developed based on ALS with NCG [9], ALS with line search along the trend (e.g. [37, 7, 43]), ALS with extrapolation along the trend (e.g. [8, 14, 16, 18, 5, 47, 37]), and ALS with over-relaxation [26].

It seems promising to combine BCD with the method in [21] when it is detected that the iteration has fallen into a narrow valley. For example one might use the trend from a few passes of BCD to take a long linear or parabolic extrapolation step, then relax back to the floor before considering whether to accept the result. Our analysis confirms that the idea of their method should work for BCD methods, with the following caveat: it may take more than one step to relax, since one BCD step will not usually fall completely back into the valley floor. But, given that iterates convergence back to the floor linearly, the number of steps needed to descend back to the valley floor should not be large and could be estimated and controlled.

[1]Thanks to Nathaniel McClatchey for pointing this out to us.

References

[1] Evrim Acar, Daniel M. Dunlavy, and Tamara G. Kolda. A scalable optimization approach for fitting canonical tensor decompositions. *J. Chemometrics*, 25(2):67–86, Feb 2011. `doi:10.1002/cem.1335`.

[2] Amir Beck and Luba Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM J. Optim.*, 23(4):2037–2060, 2013. `doi:10.1137/120887679`.

[3] Gregory Beylkin and Martin J. Mohlenkamp. Numerical operator calculus in higher dimensions. *Proc. Natl. Acad. Sci. USA*, 99(16):10246–10251, 2002. `doi:10.1073/pnas.112329799`.

[4] Gregory Beylkin and Martin J. Mohlenkamp. Algorithms for numerical analysis in high dimensions. *SIAM J. Sci. Comput.*, 26(6):2133–2159, 2005. `doi:10.1137/040604959`.

[5] Rasmus Bro. *Multi-way Analysis in the Food Industry: Models, Algorithms, and Applications*. PhD thesis, Universiteit van Amsterdam, Amsterdam, November 1998.

[6] Bilian Chen, Simai He, Zhening Li, and Shuzhong Zhang. Maximum block improvement and polynomial optimization. *SIAM J. Optim.*, 22(1):87–107, 2012. `doi:10.1137/110834524`.

[7] P. Comon, X. Luciani, and A. L. F. de Almeida. Tensor decompositions, alternating least squares and other tales. *J. Chemometrics*, 23(7-8):393–405, Jul-Aug 2009. `doi:10.1002/cem.1236`.

[8] H. De Sterck. A nonlinear GMRES optimization algorithm for canonical tensor decomposition. *SIAM J. Sci. Comput.*, 34(3):A1351–A1379, 2012. `doi:10.1137/110835530`.

[9] Hans De Sterck and Manda Winlaw. A nonlinearly preconditioned conjugate gradient algorithm for rank-R canonical tensor approximation. *Numer. Linear Algebra Appl.*, 22(3):410–432, 2015. `doi:10.1002/nla.1963`.

[10] Mike Espig, Wolfgang Hackbusch, Thorsten Rohwedder, and Reinhold Schneider. Variational calculus with sums of elementary tensors of fixed rank. *Numer. Math.*, 122(3):469–488, 2012. `doi:10.1007/s00211-012-0464-x`.

[11] R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *Comput. J.*, 7:149–154, 1964. `doi:10.1093/comjnl/7.2.149`.

[12] Gene H. Golub and Dianne P. O'Leary. Some history of the conjugate gradient and Lanczos algorithms: 1948–1976. *SIAM Rev.*, 31(1):50–102, 1989. `doi:10.1137/1031003`.

[13] X. Gong, M. Mohlenkamp, and T. Young. The optimization landscape for fitting a rank-2 tensor with a rank-1 tensor. *SIAM Journal on Applied Dynamical Systems*, 17(2):1432–1477, 2018. `doi:10.1137/17M112213X`.

[14] R. A. Harshman. Foundations of the PARAFAC procedure: Model and conditions for an "explanatory" multi-mode factor analysis. Working Papers in Phonetics 16, UCLA, Los Angeles, 1970. URL: `http://www.psychology.uwo.ca/faculty/harshman/wpppfac0.pdf`.

[15] Mingyi Hong, Meisam Razaviyayn, Zhi-Quan Luo, and Jong-Shi Pang. A Unified Algorithmic Framework for Block-Structured Optimization Involving Big Data. *IEEE Signal Process. Mag.*, 33(1):57–77, Jan 2016. `doi:10.1109/MSP.2015.2481563`.

[16] PK Hopke, P Paatero, H Jia, RT Ross, and RA Harshman. Three-way (PARAFAC) factor analysis: examination and comparison of alternative computational methods as applied to ill-conditioned data. *Chemometrics Intell. Lab. Syst.*, 43(1-2):25–42, Sep 28, 1998. `doi:10.1016/S0169-7439(98)00077-X`.

[17] Pieter M. Kroonenberg and Jan de Leeuw. Principal component analysis of three-mode data by means of alternating least squares algorithms. *Psychometrika*, 45(1):69–97, 1980. `doi:10.1007/BF02293599`.

[18] S. E. Leurgans, R. T. Ross, and R. B. Abel. A decomposition for three-way arrays. *SIAM J. Matrix Anal. Appl.*, 14(4):1064–1083, 1993. `doi:10.1137/0614071`.

[19] Na Li. *Variants of ALS on Tensor Decompositions and Applications*. PhD thesis, Clarkson University, Potsdam, NY, 2013.

[20] Na Li, Stefan Kindermann, and Carmeliza Navasca. Some convergence results on the regularized alternating least-squares method for tensor decomposition. *Linear Algebra Appl.*, 438(2):796–812, 2013. `doi:10.1016/j.laa.2011.12.002`.

[21] Qing-Xiao Li, Rong-Qiang He, and Zhong-Yi Lu. Accelerating optimization by tracing valley. *Comput. Phys. Commun.*, 203:168–177, 2016. `doi:10.1016/j.cpc.2016.03.002`.

[22] David G. Luenberger and Yinyu Ye. *Linear and nonlinear programming*, volume 228 of *International Series in Operations Research & Management Science*. Springer, Cham, fourth edition, 2016. `doi:10.1007/978-3-319-18842-3`.

[23] Z. Q. Luo and P. Tseng. On the convergence of the coordinate descent method for convex differentiable minimization. *J. Optim. Theory Appl.*, 72(1):7–35, 1992. `doi:10.1007/BF00939948`.

[24] Dhruv Mahajan, S. Sathiya Keerthi, and S. Sundararajan. A distributed block coordinate descent method for training l_1 regularized linear classifiers. *J. Mach. Learn. Res.*, 18:Paper No. 91, 35, 2017.

[25] Jiří Matoušek. *Lectures on discrete geometry*, volume 212 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2002. `doi:10.1007/978-1-4613-0039-7`.

[26] Nathaniel McClatchey. *Tensors: an Adaptive Approximation Algorithm, Convergence in Direction, and Connectedness Properties*. PhD thesis, Ohio University, 2018.

[27] Martin J. Mohlenkamp. Musings on multilinear fitting. *Linear Algebra Appl.*, 438(2):834–852, 2013. `doi:10.1016/j.laa.2011.04.019`.

[28] Martin J. Mohlenkamp. The dynamics of swamps in the tensor approximation problem. Submitted for publication, 2018. URL: `http://www.ohiouniversityfaculty.com/mohlenka/research/MOHLEN2018p.pdf`.

[29] Carmeliza Navasca, Lieven De Lathauwer, and Stefan Kinderman. Swamp reducing technique for tensor decomposition. In *16th European Signal Processing Conference (EUSIPCO 2008)*, Lausanne, Switzerland, 2008. European Association for Signal Processing. URL: `http://ieeexplore.ieee.org/document/7080724/`.

[30] Yu. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM J. Optim.*, 22(2):341–362, 2012. `doi:10.1137/100802001`.

[31] P Paatero. A weighted non-negative least squares algorithm for three-way 'parafac' factor analysis. *Chemometrics Intell. Lab. Syst.*, 38(2):223–242, Oct 1997. 2nd Internet Conference in Chemometrics (InCINC), Potsdam, NY, 1996. `doi:10.1016/S0169-7439(97)00031-2`.

[32] P Paatero. Construction and analysis of degenerate PARAFAC models. *J. Chemometrics*, 14(3):285–299, May-Jun 2000.
`doi:10.1002/1099-128X(200005/06)14:3<285::AID-CEM584>3.3.CO;2-T`.

[33] Pentti Paatero. The multilinear engine—a table-driven, least squares program for solving multilinear problems, including the n-way parallel factor analysis model. *J. Comput. Graph. Statist.*, 8(4):854–888, 1999. `doi:10.2307/1390831`.

[34] M. J. D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Comput. J.*, 7:155–162, 1964.
`doi:10.1093/comjnl/7.2.155`.

[35] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical recipes*. Cambridge University Press, Cambridge, 1986.

[36] Zhiwei Qin, Katya Scheinberg, and Donald Goldfarb. Efficient block-coordinate descent algorithms for the group Lasso. *Math. Program. Comput.*, 5(2):143–169, 2013.
`doi:10.1007/s12532-013-0051-x`.

[37] Myriam Rajih, Pierre Comon, and Richard A. Harshman. Enhanced line search: a novel method to accelerate PARAFAC. *SIAM J. Matrix Anal. Appl.*, 30(3):1128–1147, 2008.
`doi:10.1137/06065577`.

[38] William S. Rayens and Menjamin C. Mitchell. Two-factor degeneracies and a stabilization of PARAFAC. *Chemometrics Intell. Lab. Syst.*, 38:173–181, 1997.

[39] Meisam Razaviyayn, Mingyi Hong, and Zhi-Quan Luo. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM J. Optim.*, 23(2):1126–1153, 2013. `doi:10.1137/120891009`.

[40] Matthew J. Reynolds, Alireza Doostan, and Gregory Beylkin. Randomized alternating least squares for canonical tensor decompositions: application to a PDE with random data. *SIAM J. Sci. Comput.*, 38(5):A2634–A2664, 2016. `doi:10.1137/15M1042802`.

[41] Peter Richtárik and Martin Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Math. Program.*, 144(1-2, Ser. A):1–38, 2014. `doi:10.1007/s10107-012-0614-z`.

[42] H. H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *Comput. J.*, 3:175–184, 1960/1961. doi:10.1093/comjnl/3.3.175.

[43] Laurent Sorber, Ignat Domanov, Marc Van Barel, and Lieven De Lathauwer. Exact line and plane search for tensor optimization. *Comput. Optim. Appl.*, 63(1):121–142, 2016. doi:10.1007/s10589-015-9761-5.

[44] Laurent Sorber, Marc Van Barel, and Lieven De Lathauwer. Optimization-based algorithms for tensor decompositions: canonical polyadic decomposition, decomposition in rank-$(L_r, L_r, 1)$ terms, and a new generalization. *SIAM J. Optim.*, 23(2):695–720, 2013. doi:10.1137/120868323.

[45] Jos M. F. ten Berge. *Least squares optimization in multivariate analysis*, volume 25 of *M&T Series*. D.S.W.O. Press, Leiden, The Netherlands, 1993.

[46] Petr Tichavsky, Anh-Huy Phan, and Andrzej Cichocki. Partitioned Alternating Least Squares Technique for Canonical Polyadic Tensor Decomposition. *IEEE Signal Process. Lett.*, 23(7):993–997, Jul 2016. doi:10.1109/LSP.2016.2577383.

[47] Giorgio Tomasi and Rasmus Bro. A comparison of algorithms for fitting the PARAFAC model. *Comput. Statist. Data Anal.*, 50(7):1700–1734, 2006. doi:10.1016/j.csda.2004.11.013.

[48] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *J. Optim. Theory Appl.*, 109(3):475–494, 2001. doi:10.1023/A:1017501703105.

[49] Wu-Hsiung Tung, Tien-Tso Lee, Weng-Sheng Kuo, and Shung-Jung Yaur. Optimization of axial enrichment distribution for BWR fuels using scoping libraries and block coordinate descent method. *Nucl. Eng. Des.*, 313:84–95, Mar 2017. doi:10.1016/j.nucengdes.2016.12.003.

[50] André Uschmajew. Local convergence of the alternating least squares algorithm for canonical tensor approximation. *SIAM J. Matrix Anal. Appl.*, 33(2):639–652, 2012. doi:10.1137/110843587.

[51] André Uschmajew. A new convergence proof for the higher-order power method and generalizations. *Pac. J. Optim.*, 11(2):309–321, 2015.

[52] J. Warga. Minimizing certain convex functions. *J. Soc. Indust. Appl. Math.*, 11:588–593, 1963. doi:10.1137/0111043.

[53] Yangyang Xu and Wotao Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM J. Imaging Sci.*, 6(3):1758–1789, 2013. doi:10.1137/120887795.