

ABSTRACT

PROCESSING AND INPAINTING OF SPARSE DATA AS APPLIED TO  
ATOMIC FORCE MICROSCOPY IMAGING

By

Rodrigo Bouchardet Farnham

August 2012

When data collection is expensive, gathering fewer and strategically located points may reduce costs while maintaining important information. Inpainting then allows for the intelligent reconstruction of missing data from the sparse observations. In this thesis work, we propose the least squares differences algorithm, a new scheme for de-trending data with repeated observations based on classical least squares fitting along with an empirical guideline for constructing good sampling strategies. Furthermore, we develop and present a novel inpainting algorithm – penalized dictionary inpainting – that utilizes a variable penalty term and exhibits nonlocal sensitivity. Armed with these two innovations, we illustrate how current atomic force microscopy (AFM) imaging can be made more efficient. In particular, we apply least squares differences to the analysis of non-raster scanning methodologies, along with the inpainting of sparse or subsampled data.

PROCESSING AND INPAINTING OF SPARSE DATA AS APPLIED TO  
ATOMIC FORCE MICROSCOPY IMAGING

A THESIS

Presented to the Department of Mathematics and Statistics  
California State University, Long Beach

In Partial Fulfillment  
of the Requirements for the Degree  
Master of Science in Applied Mathematics

Committee Members:

Jen-Mei Chang, Ph.D.  
Eun Heui Kim, Ph.D.  
Joseph Bennish, Ph.D.

College Designee:

Robert Mena, Ph.D.

By Rodrigo Bouchardet Farnham

B.S. Mathematics, 2009, Louisiana State University

August 2012

WE, THE UNDERSIGNED MEMBERS OF THE COMMITTEE,  
HAVE APPROVED THIS THESIS

PROCESSING AND INPAINTING OF SPARSE DATA AS APPLIED TO  
ATOMIC FORCE MICROSCOPY IMAGING

By

Rodrigo Bouchardet Farnham

COMMITTEE MEMBERS

---

Jen-Mei Chang, Ph.D.

Mathematics & Statistics

---

Eun Heui Kim, Ph.D.

Mathematics & Statistics

---

Joseph Bennish, Ph.D.

Mathematics & Statistics

ACCEPTED AND APPROVED ON BEHALF OF THE UNIVERSITY

---

Robert Mena, Ph.D.

Department Chair, Department of Mathematics and Statistics

California State University, Long Beach

August 2012

## TABLE OF CONTENTS

	Page
LIST OF TABLES .....	v
LIST OF FIGURES .....	vi
CHAPTER	
1. INTRODUCTION .....	1
2. THEORY OF DIFFERENCES .....	3
Classical Least Squares.....	3
Least Squares Differences .....	8
Numerical Stability of Solution.....	12
Summary .....	14
3. MORE ON BASES .....	15
Polynomial and Trigonometric Bases .....	15
B-Splines .....	16
Smoothing Splines: Adding a Penalty Term.....	19
Efficiently Evaluating B-Splines .....	22
Summary .....	23
4. INPAINTING .....	25
Netflix Prize .....	25
Penalized Dictionary Inpainting .....	31
Summary .....	34
5. EXPERIMENTAL RESULTS .....	39
Atomic Force Microscopy .....	39
Raster Scan And Line Flattening .....	40
Alternative Scan Paths.....	43
Inpainting And Sparsity .....	45
Putting It All Together .....	48
Summary .....	50
6. SUMMARY AND CONCLUSIONS .....	52
Least Squares Differences .....	52
Penalized Dictionary Inpainting .....	52

	Page
Future Work .....	52
SOURCE CODE .....	54
BIBLIOGRAPHY .....	67

## LIST OF TABLES

TABLE		Page
1.	Computational Complexity of B-Spline Evaluation (seconds) .....	24
2.	Errors for the first twenty iterations of the Netflix Algorithm.....	30
3.	Convergence of PDI .....	38

## LIST OF FIGURES

FIGURE		Page
1.	Blow up for a polynomial fit .....	17
2.	Illustration of a few different B-Spline bases elements .....	18
3.	The effect of the inclusion of phantom knots .....	19
4.	The effect of regularization on B-Spline Regression .....	22
5.	Convergence of error after many iterations .....	28
6.	The Netflix Algorithm repairing a damaged pattern .....	29
7.	The box highlights a typical neighborhood of an image .....	33
8.	The repeating pattern corresponds to rows in the image .....	35
9.	Penalized dictionary inpainting pattern completion with different $\lambda$ s .	37
10.	Atomic force microscopy schematic.....	41
11.	Line flattening of a raster scan with thermal drift .....	42
12.	Distortions are introduced when assumptions do not hold .....	43
13.	Cycloid scan path with its associated bundle plot .....	44
14.	Spiral scan path with its associated bundle plot .....	45

FIGURE	Page
15. Image generated from a simulated cycloid scan sans drift correction ..	46
16. Image generated from a simulated cycloid scan after drift correction..	47
17. Comparison between PDI (left) and TV-L1 (right) .....	48
18. A screenshot demonstrating PDI's slow convergence.....	49
19. Cycloid scans of calibration grids.....	50
20. Before and after drift correction images of spiral scanned gold .....	51



## CHAPTER 1

### INTRODUCTION

The motivation for our work came from the current challenges faced in the field of atomic force microscopy (AFM). However, the theory we developed is sufficiently general to be of use in other problem domains. For this reason, the paper is comprised of two independent parts.

The first part of this paper is devoted to developing two distinct mathematical algorithms free of context. The hope is that these novel algorithms may prove useful in other applications, if not fascinating by their own right. Least squares differences is our extension of classical least squares fitting, suitable to the analysis of data with repeated, “overlapping” observations. By leveraging a somewhat unusual basis and a little linear algebra, we show how to effectively model any low frequency noise that may be present in signals, provided a few criteria are met. Penalized dictionary inpainting, on the other hand, is our own attempt at creating an inpainting algorithm with the ability to complete patterns and edges, in addition to propagating known information into unknown territory.

The second part is concerned with applications to atomic force microscopy, and how the two algorithms play relevant roles in the cutting-edge of fast AFM. Atomic force microscopy is a relatively new imaging technique which sees, not through light, but by the tactile response of a microscopic probe upon the substrate. Atomic force microscopes are in the cutting-edge for imaging and manipulating matter in the small scale. However, there is still much to be improved in the field, especially in regards to imaging speed. To that end we adopt

alternative scanning techniques which are more auspicious for quick scans, and these require more advanced analytical techniques than have been available so far to the field of atomic force microscopy. Least squares differences plays the part of background removal, rectifying an otherwise stilted signal, while penalized dictionary inpainting allows us to infer what has gone unseen. In practice, we found that least squares differences performs very well when applied to real data collected by real AFMs. However, penalized dictionary inpainting runs much too slowly to be an alternative for the already established methods of inpainting. Nonetheless, the combination of non-raster scan paths with the least squares differences algorithm and sparse image inpainting proves to be a competitive alternative to the current methods. Furthermore, unlike raster scans, non-raster scans are amenable to a direct scaling of scan speeds, so that any improvements in hardware capability will directly translate to workable gains under the non-raster, least squares differences, and inpainting framework.

## CHAPTER 2

### THEORY OF DIFFERENCES

This chapter is aimed at developing our theory of differences, which is a natural extension of classical least squares fitting. It is of particular practical importance because it is readily applied to observations which have a certain degree of redundancy. First, we present and review some results about regular least squares fitting, then we mirror the development with differences.

As applied to our work in atomic force microscopy, the algorithm of least squares differences developed in this chapter is the cornerstone of our method for removing deleterious background noise and thermal drift. Concretely, our aim is to leverage the “overlapping” observations of alternative scan paths to model the background noise accurately.

#### Classical Least Squares

In this section we define some useful terms and give a brief introduction to the classical theory of least squares approximation.

**Definition 2.1.** *Given a function  $f : \Omega \rightarrow \mathbb{R}$ , we call some finite collection of pairs  $(\mathbf{x}, f(\mathbf{x}))$  a set of observations  $\mathcal{O}$ .*

**Theorem 2.2.** *Let  $\Omega \subset \mathbb{R}$ . Given observations  $\mathcal{O}$  of a function  $f : \Omega \rightarrow \mathbb{R}$  and a basis  $\{\phi_n\}$  on  $\Omega$ , we can construct a finite dimensional least squares approximation,  $\tilde{f} = \sum_{i=1}^n \alpha_i \phi_i$  by solving a linear system for the coefficients  $\alpha$  [1].*

*Proof.* Let

$$\mathbf{F} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_l) \end{bmatrix}, \Phi = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_n(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \cdots & \phi_n(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(x_l) & \phi_2(x_l) & \cdots & \phi_n(x_l) \end{bmatrix}, \alpha = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix}$$

where  $\mathcal{O} = \{(x_k, f(x_k))\}$ .

Then define the squared error of an approximation by  $E = \|\mathbf{F} - \Phi\alpha\|^2$ . We wish to find the choice of  $\alpha$  which minimizes the error.

$$E = (\mathbf{F} - \Phi\alpha)^T(\mathbf{F} - \Phi\alpha) = \mathbf{F}^T\mathbf{F} - 2\alpha^T\Phi^T\mathbf{F} + \alpha^T\Phi^T\Phi\alpha. \quad (2.1)$$

$$\frac{\partial E}{\partial \alpha} = -2\Phi^T\mathbf{F} + 2\Phi^T\Phi\alpha = 0 \quad (2.2)$$

$$\Phi^T\mathbf{F} = \Phi^T\Phi\alpha \quad (2.3)$$

$$\alpha = (\Phi^T\Phi)^{-1}\Phi^T\mathbf{F}. \quad (2.4)$$

□

Equation 2.4 above has a solution minimizing  $E$  provided  $\Phi^T\Phi$  is nonsingular. Usually this condition is satisfied if there are sufficient observations; that is, more observations than basis elements. However, unusual circumstances may arise which cause even an excess of observations to have many linear dependencies, rendering the matrix  $\Phi^T\Phi$  rank deficient. In such singular cases, one can still recover some form of solution by expressing  $\Phi$  in its singular value decomposition, making special use of the property that both  $\mathbf{U}$  and  $\mathbf{V}$  are orthogonal matrices.

$$\Phi\mathbf{F} = \Phi^T\Phi\alpha, \Phi = \mathbf{U}\mathbf{S}\mathbf{V}^T$$

$$\mathbf{V}\mathbf{S}^T\mathbf{U}^T\mathbf{F} = \mathbf{V}\mathbf{S}^T\mathbf{S}\mathbf{V}^T\alpha$$

$$\alpha = \mathbf{V}\mathbf{S}^{-1}\mathbf{U}^T\mathbf{F}$$

where  $\mathbf{S}^{-1}$  is the element-wise inverse of the nonzero diagonal elements of  $S$ .

Notice that the procedure detailed above is basis-agnostic, and as such is applicable to any set of basis. Common examples are the polynomial basis  $\{x^n\}$  and the trigonometric basis  $\{\sin nx, \cos nx\}$ .

**Corollary 2.3.** (Linear Regression) *Let  $\Omega \subset \mathbb{R}$ . Given observations  $\mathcal{O}$  of a function  $f : \Omega \rightarrow \mathbb{R}$ , we can construct a linear least squares approximation to  $f$  by solving a linear system for the coefficients  $\alpha$  [1].*

*Proof.* This is a simple application of Theorem 2.2 with  $\Phi = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_l \end{bmatrix}$  □

**Theorem 2.4.** *Let  $f : \Omega \rightarrow \mathbb{R}$ . Given an orthonormal basis  $\{\phi_n\}$  on  $\Omega$ , the least squares approximation of  $f$  is equivalent to the orthogonal projection of  $f$  onto  $\{\phi_n\}$  [7].*

*Proof.* Define the least squares error by

$$E = \int_{\Omega} (f - \sum_n \alpha_n \phi_n)^2 d\mu.$$

$$\frac{\partial E}{\partial \alpha_j} = -2 \int_{\Omega} (f - \sum_n \alpha_n \phi_n) \phi_j d\mu.$$

From the minimization condition  $\frac{\partial E}{\partial \alpha_j} = 0$  we obtain

$$\int_{\Omega} f \phi_j d\mu = \int_{\Omega} \sum_n \alpha_n \phi_n \phi_j d\mu = \sum_n \alpha_n \int_{\Omega} \phi_n \phi_j d\mu.$$

Then orthonormality of our basis ensures that all but one of the terms on the right vanish.

$$\int_{\Omega} f \phi_j d\mu = \alpha_j.$$

Now recognize that  $\int_{\Omega} f \phi_j d\mu = \langle f, \phi_j \rangle$  is the projection of  $f$  onto the  $j^{th}$  basis element. □

In the theorem above we did not require that the region of interest  $\Omega$  be a finite set of discrete points, or indeed even countable. To apply Theorem 2.4 to a finite or countable set of points, you need only use the discrete measure in place of  $\mu$ .

**Corollary 2.5.** *Given observations  $\mathcal{O}$  of a function  $f : \Omega \rightarrow \mathbb{R}$ , and a basis  $\{\phi_n\}$  that is orthonormal on the domain of observations, the least squares approximation of  $f$  is the orthogonal projection of  $f$  onto  $\{\phi_n\}$  [7].*

*Proof.* Apply Theorem 2.4 with  $\Omega$  being the domain of observations and  $\mu$  the discrete measure. □

Corollary 2.5 is not as useful as it may seem. Although computing projections is a lot less costly than solving the linear system of Theorem 2.2, the crucial hypothesis that  $\{\phi_n\}$  be orthonormal on the domain of observations  $\mathcal{O}$  is, in general, not true. For example, although the trigonometric basis is orthogonal on the interval  $[0, \frac{\pi}{2}]$ , it is not necessarily so on an arbitrary collection of discrete points. However, as we see below, given an orthonormal basis on some domain  $\mathbf{X}$ , it is possible to construct a related basis that is orthonormal on a restriction of the domain. This basis can then be used repeatedly to quickly find the least squares approximation of many different functions, so long as we only care about the restricted domain.

**Definition 2.6.** *Given a basis  $\{\phi_n\}$  on a space  $\mathbf{X}$  and a subset  $\Omega \subset \mathbf{X}$ , we define the restricted basis  $\{\phi_n^{\Omega}\}$  by restricting the domain of each basis element to  $\Omega$ . The original basis  $\{\phi_n\}$  is called the mother basis.*

Gram-Schmidt Orthogonalization allows us to construct an orthonormal restricted basis while retaining some of the qualities of its mother basis, such as

---

**Algorithm 1** Gram-Schmidt Orthogonalization [7]

---

1. for  $i = 0, 1, \dots, m$

(a) Set  $\widehat{\phi}_i^\Omega$  to  $\phi_i$

(b) for  $j = 0, 1, \dots, i - 1$

i. Subtract the (restricted) projection of  $\widehat{\phi}_j^\Omega$  from  $\widehat{\phi}_i^\Omega$

ii. Set  $\widehat{\phi}_i^\Omega$  to the remainder of the difference above

2. Normalize  $\widehat{\phi}_i^\Omega$

---

continuity or differentiability. This is of importance as we will see later. Armed with a restricted orthonormal basis we can perform least squares fitting as easily as computing inner products, as Corollary 2.5 describes. However, there is one caveat while performing Gram-Schmidt Orthogonalization on a restricted basis. While by definition the mother basis is linearly independent, its restriction may not be. A trivial example is some basis  $\{\phi_n\}$  restricted to  $\Omega = \{0\}$ . Clearly the restriction can have dimension at most 1. This pathology manifests itself in the Gram-Schmidt algorithm when normalizing. Should some basis element  $\phi_i^\Omega$  be a linear combination of the previous  $\{\phi_n^\Omega\}_{n=1}^{i-1}$  it can be safely discarded, as it has no additional discriminative power in the restricted space.

**Example 1.**

$$\text{Let } \{\phi_n\} = \{\sin x, \sin 2x, \sin 3x\} \text{ and } \Omega = \left\{\frac{\pi}{3}, \frac{\pi}{2}\right\}$$

*Then we can write our restricted basis succinctly like so*

$$\{\phi_n^\Omega\} = \left\{\left[\frac{\sqrt{3}}{2}, 1\right], \left[\frac{\sqrt{3}}{2}, 0\right], [0, -1]\right\}$$

*Applying algorithm the Gram-Schmidt algorithm, we produce a new basis that is*

orthonormal on the restricted domain  $\Omega$ .

$$\widehat{\phi}_1^\Omega = \frac{\phi_1^\Omega}{\|\phi_1^\Omega\|} = \left[ \frac{\sqrt{3}}{\sqrt{7}}, \frac{2}{\sqrt{7}} \right]$$

$$\widehat{\phi}_2^\Omega = \frac{\phi_2^\Omega - \langle \phi_2^\Omega, \widehat{\phi}_1^\Omega \rangle \widehat{\phi}_1^\Omega}{\|\phi_2^\Omega - \langle \phi_2^\Omega, \widehat{\phi}_1^\Omega \rangle \widehat{\phi}_1^\Omega\|} = \left[ \frac{2}{\sqrt{7}}, -\frac{\sqrt{3}}{\sqrt{7}} \right]$$

But now the process must stop, because  $\phi_3^\Omega$  is by necessity linearly dependent upon the two previous vectors:

$$\phi_3^\Omega - \langle \phi_3^\Omega, \widehat{\phi}_1^\Omega \rangle \widehat{\phi}_1^\Omega - \langle \phi_3^\Omega, \widehat{\phi}_2^\Omega \rangle \widehat{\phi}_2^\Omega = [0, 0]$$

□

The previous example illustrates that the linear independence of a mother basis does not guarantee the linear independence of its restricted basis. In practice, this should not pose any problem provided  $\Omega$  has sufficiently many points. However,  $\Omega$  may be much too small to force independence or its very structure can conspire against orthogonality of the restricted basis. If so, the system of Theorem 2.2 will be singular. In particular any minimizing solution will be but one member of an infinite family of solutions. In such cases, it is possible to cull the restricted basis by way of Gram-Schmidt Orthogonalization rather than resort to the pseudo-inverse solution detailed before.

### Least Squares Differences

We now begin the development of our differences method in earnest. Many of the results should seem familiar, as they mimic the exposition given in Section 2.1.

**Definition 2.7.** *Given a function  $f : \mathbf{X} \rightarrow \mathbb{R}$ , we call a finite collection of triples  $(\mathbf{a}, \mathbf{b}, f(\mathbf{a}) - f(\mathbf{b}))$ , where  $\mathbf{a}$  and  $\mathbf{b}$  are elements of  $\mathbf{X}$ , a set of difference observations  $\mathcal{O}$ , or differences for short.*



Note how this is a special case of Definition 2.1. We could study the related function  $F : \mathbf{X} \times \mathbf{X} \rightarrow \mathbb{R}$  and consider our observations to be  $\mathcal{O}'$  with domain  $\mathbf{X} \times \mathbf{X}$  and values  $F(\mathbf{x}, \mathbf{y}) = f(\mathbf{x}) - f(\mathbf{y})$ . However, it will be convenient later to think of difference observations separately.

**Theorem 2.8.** (Least Squares Differences) *Let  $\Omega \subset \mathbb{R}$ . Given difference observations  $\mathcal{O}$  of a function  $f : \Omega \rightarrow \mathbb{R}$  and a basis  $\{\phi_n\}$  on  $\Omega$ , we can construct a finite dimensional least squares approximation,  $\tilde{f} = \sum_n \alpha_i \phi_i$  by solving a linear system.*

*Proof.* Notice that  $\tilde{f}(a) - \tilde{f}(b) = \sum_n \alpha_i (\phi_i(a) - \phi_i(b))$ . By interpreting each difference observation as a constraint of the form  $f(a) - f(b) \approx \tilde{f}(a) - \tilde{f}(b)$ , we proceed much the same way as we did in Theorem 2.2.

Let

$$\mathbf{F} = \begin{bmatrix} f(a_1) - f(b_1) \\ f(a_2) - f(b_2) \\ \vdots \\ f(a_l) - f(b_l) \end{bmatrix}, \Phi = \begin{bmatrix} \phi_1(a_1) - \phi_1(b_1) & \phi_2(a_1) - \phi_2(b_1) & \cdots & \phi_n(a_1) - \phi_n(b_1) \\ \phi_1(a_2) - \phi_1(b_2) & \phi_2(a_2) - \phi_2(b_2) & \cdots & \phi_n(a_2) - \phi_n(b_2) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_1(a_l) - \phi_1(b_l) & \phi_2(a_l) - \phi_2(b_l) & \cdots & \phi_n(a_l) - \phi_n(b_l) \end{bmatrix}$$

$$\alpha = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} \text{ where } \mathcal{O} = \{(a_k, b_k, f(a_k) - f(b_k))\}.$$

Then define the squared error of an approximation by  $E = \|\mathbf{F} - \Phi\alpha\|^2$ . As before, we wish to find the choice of  $\alpha$  which minimizes the error.

$$E = (\mathbf{F} - \Phi\alpha)^T (\mathbf{F} - \Phi\alpha) = \mathbf{F}^T \mathbf{F} - 2\alpha^T \Phi^T \mathbf{F} + \alpha^T \Phi^T \Phi \alpha.$$

$$\frac{\partial E}{\partial \alpha} = -2\Phi^T \mathbf{F} + 2\Phi^T \Phi \alpha = 0$$

$$\Phi^T \mathbf{F} = \Phi^T \Phi \alpha, \alpha = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{F}.$$

□

All the caveats and remarks from Theorem 2.2 apply. In addition to the parallels, it is important to notice the differences between the two. We could have performed a least squares fit, in the vein of Theorem 2.2, to the related function  $F : \mathbf{X} \times \mathbf{X} \rightarrow \mathbb{R}$ . But we specifically chose to use a basis on the original space  $\mathbf{X}$  as it maintains the differences viewpoint, making Theorem 2.8 much more applicable.

**Definition 2.9.** A basis  $\{\phi_n\}$  of some function space  $\mathbf{F}[\mathbf{X}]$  is called “zero average” if  $\int_{\mathbf{X}} \phi_n dx = 0$  for every  $n$ .

**Theorem 2.10.** (Basis Projection Theorem for Differences) Let  $f : \Omega \rightarrow \mathbb{R}$ . Given an orthonormal and zero average basis  $\{\phi_n\}$  on  $\Omega$ , the least squares approximation of the differences of  $f$  is proportional to the orthogonal projection of  $f(x) - f(y)$  onto  $\{\phi_n(x) - \phi_n(y)\}$ .

*Proof.* For convenience, we define the differences of  $f$  by

$$F(x, y) = f(x) - f(y)$$

Then the least squares error is given by the formulation

$$E = \iint_{\Omega \times \Omega} \left[ F(x, y) - \sum_n \alpha_n (\phi_n(x) - \phi_n(y)) \right]^2 d\mu_x d\mu_y.$$

$$\frac{\partial E}{\partial \alpha_j} = -2 \iint_{\Omega \times \Omega} \left[ F(x, y) - \sum_n \alpha_n (\phi_n(x) - \phi_n(y)) \right] (\phi_j(x) - \phi_j(y)) d\mu_x d\mu_y.$$

From the minimization condition  $\frac{\partial E}{\partial \alpha_j} = 0$  we obtain

$$\begin{aligned} & \iint_{\Omega \times \Omega} F(x, y) (\phi_j(x) - \phi_j(y)) d\mu_x d\mu_y \\ &= \iint \sum_n \alpha_n (\phi_n(x) - \phi_n(y)) (\phi_j(x) - \phi_j(y)) d\mu_x d\mu_y \end{aligned}$$

$$\begin{aligned}
&= \sum_n \alpha_n \iint (\phi_n(x) - \phi_n(y))(\phi_j(x) - \phi_j(y)) d\mu_x d\mu_y \\
&= \sum_n \alpha_n \iint \phi_n(x)\phi_j(x) + \phi_n(y)\phi_j(y) - \phi_n(x)\phi_j(y) - \phi_n(y)\phi_j(x) d\mu d\mu \\
&= \sum_n \alpha_n \int_{\Omega} \langle \phi_n, \phi_j \rangle d\mu + \int_{\Omega} \langle \phi_n, \phi_j \rangle d\mu - \int_{\Omega} \phi_n d\mu \int_{\Omega} \phi_j d\mu - \int_{\Omega} \phi_n d\mu \int_{\Omega} \phi_j d\mu
\end{aligned}$$

Then the orthonormality and balance of our basis ensures that all but two of the terms on the right vanish. Thus, we arrive at

$$\iint_{\Omega \times \Omega} F(x, y)(\phi_j(x) - \phi_j(y)) d\mu_x d\mu_y = 2\alpha_j \mu(\Omega)$$

Now recognize that  $\iint_{\Omega \times \Omega} F(x, y)(\phi_j(x) - \phi_j(y)) d\mu_x d\mu_y = \langle F, \phi_j(x) - \phi_j(y) \rangle_{\Omega \times \Omega}$ , the projection of  $F$  onto the  $j^{th}$  difference basis element.

$$\alpha_j = \frac{\langle F, \phi_j(x) - \phi_j(y) \rangle_{\Omega \times \Omega}}{2\mu(\Omega)}$$

□

**Corollary 2.11.** *Given difference observations  $\mathcal{O}$  of a function  $f : \Omega \rightarrow \mathbb{R}$ , and a basis  $\{\phi_n\}$  that is orthonormal and zero average on the domain of observations, the least squares approximation of  $f$  is proportional to the orthogonal projection of  $F(x, y) = f(x) - f(y)$  onto  $\{\phi_n(x) - \phi_n(y)\}$ .*

*Proof.* Apply Theorem 2.10 with  $\Omega$  being the domain of observations and  $\mu$  the discrete measure. □

Corollary 2.11 enables us to quickly perform least squares fitting of difference observations, provided we have at our disposal a orthonormal and zero average basis on the domain of observations, since under those circumstances the least squares differences fit can be done by a simple inner product. This is often not the case, usual bases, such as the polynomial and trigonometric, are not in general orthogonal nor zero average on a restricted set. We can overcome this

---

**Algorithm 2** Augmented Gram-Schmidt Orthogonalization

---

For each restricted basis element, first balance them by subtracting their integral divided by the measure of the space. Then proceed with Gram-Schmidt as normally.

---

problem, much in the same way we did before. Example 2 demonstrates why the augmented Gram-Schmidt process preserves zero average.

**Example 2.** *Let  $\{\phi_n^\Omega\}$  be some restricted basis on  $\Omega \subset \mathbf{X}$ . We construct an orthonormal and zero average basis on  $\Omega$  as follows:*

*Define*

$$\psi_n^\Omega = \phi_n^\Omega - c_n$$

*where  $c_n = \frac{1}{\mu(\Omega)} \int_\Omega \phi_n^\Omega d\mu$ , so that  $\psi_n^\Omega$  is zero average. Then notice that any linear combination of zero average basis elements is still zero average:*

$$\int_\Omega \sum_n \beta_n \psi_n^\Omega d\mu = \sum_n \int_\Omega \beta_n \psi_n^\Omega d\mu = 0$$

*So in particular, Gram-Schmidt Orthogonalization will preserve the zero average aspect of the basis.* □

We can use Corollary 2.11 and the Augmented Gram-Schmidt Orthogonalization algorithm to quickly perform least squares differences fitting on data with the same observation domains. That is, by making the investment of orthogonalizing a (possibly restricted) basis on some domain, all subsequent least squares differences fit on that domain can be done via inner products, which is asymptotically faster than solving a linear system. This is both of use in practice and of theoretical interest.

### Numerical Stability of Solution

While the previous sections were mostly concerned with a theoretical

formulation for the theory of differences, little mention was given to numerical stability. In fact, the proposed solution involving the normal equation is suboptimal due to its inherent numerical instability – the inversion of a possibly ill-conditioned matrix.

**Example 3.** *Let*

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & \epsilon \end{bmatrix}$$

*So that  $\det(\mathbf{A}) = \epsilon$ .*

*But,*

$$\mathbf{A}^T \mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & \epsilon^2 \end{bmatrix}$$

*with  $\det(\mathbf{A}^T \mathbf{A}) = \epsilon^2$ . Therefore the symmetric matrix  $\mathbf{A}^T \mathbf{A}$  is more ill-conditioned than the original matrix  $\mathbf{A}$ .*

Example 3 suggests that the normal equation solution to the least squares problem detailed before is numerically unstable, since it involves the inverse of the more ill-conditioned matrix  $\Phi^T \Phi$ . Fortunately, the least squares problem can be solved directly, without ever forming the ill-conditioned symmetric matrix  $\Phi^T \Phi$ .

Returning to the original formulation of the problem, we seek to find  $\alpha$  such that the error  $\|\mathbf{F} - \Phi\alpha\|^2$  is minimized. To find an alternative, more numerically stable solution, we exploit the fact that if  $\mathbf{U}$  is an orthogonal matrix,  $\|\mathbf{U}x\|^2 = \|x\|^2$ . This is readily apparent from calculation:

$$\|\mathbf{U}x\|^2 = (\mathbf{U}x)^T (\mathbf{U}x) = x^T \mathbf{U}^T \mathbf{U} x = x^T x = \|x\|^2$$

By way of singular value decomposition, we write  $\Phi = \mathbf{U} \mathbf{S} \mathbf{V}^T$ . So that

$$\|\mathbf{F} - \Phi\alpha\|^2 = \|\mathbf{U}(\mathbf{U}^T \mathbf{F} - \mathbf{S} \mathbf{V}^T \alpha)\|^2 = \|\mathbf{U}^T \mathbf{F} - \mathbf{S} \mathbf{V}^T \alpha\|^2$$

Letting  $\varphi = \mathbf{V}^T \alpha$  and  $\Delta = \mathbf{U}^T \mathbf{F}$  we cast the original problem to an equivalent formulation

$$\|\Delta - \mathbf{S}\varphi\|^2.$$

This new system can easily be minimized since  $\mathbf{S}$  is diagonal. In some cases,  $\mathbf{S}$  may be rank deficient (more columns than rows). This implies a choice for  $\varphi$ , and by convention we set all the free variables to 0, thereby minimizing  $\varphi$ 's magnitude. For the typical case, where there are many more rows than columns, we ignore the zero rows, giving rise to the residual error. After solving for  $\varphi$ , recovering  $\alpha = \mathbf{V}\varphi$  is a simple matter.

### Summary

In this chapter we laid the foundation for the analysis of differences observations. These are of practical importance in our work with AFMs. Remaining questions about choice of basis shall be addressed in the next chapter.

## CHAPTER 3

### MORE ON BASES

In Chapter 2 we discussed least squares fitting and least squares differences in a general setting. For any practical application we require that the general abstractions be made specific. In particular, Chapter 2 did not address the choice of basis – it simply assumed that one was already available. It turns out that real-world performance is intimately tied with the choice of basis. That is, the usual polynomial or trigonometric bases exhibit undesirable numerical instabilities, especially the higher order terms.

#### Polynomial and Trigonometric Bases

The most commonly used bases are the polynomial basis

$$\{x^n\}_{n=0}^{\infty}$$

and trigonometric basis

$$\{\sin nx\}_{n=1}^{\infty} \cup \{\cos nx\}_{n=0}^{\infty} \text{ or } \{e^{inx}\}_{n=-\infty}^{\infty}$$

For any computation on a real computer we must consider only finitely many terms, so when we speak of a basis in practice, we mean a finite basis spanning a subspace of the full space.

To accurately approximate an arbitrary function  $f$ , we usually require many terms. This is problematic for the two bases above, as finite precision floating point numbers lead to egregious arithmetical errors. Example 4 clearly demonstrates this limitation inherent of computer arithmetic.

**Example 4.** *Approximating a function  $f$  on the interval  $[0, 2]$  with a polynomial basis quickly becomes infeasible. Consider the points  $x_1 = .0001$  and  $x_2 = 1.9999$ .*

*Then*

$$x_1^{20} \approx 0 \text{ and } x_2^{20} \approx 1,047,528.$$

*So that, under machine arithmetic, all precision is lost when differencing the two:*

$$x_2^{20} - x_1^{20} = x_2^{20}$$

Additionally, polynomials necessarily blow up at the edges of the approximating region (see Example 5). There are ways to deal with this, such as extending the approximating region with synthetic data. But even so, numerical instability precludes any approximation relying on the detail of higher-order terms. Similar problems arise when dealing with the higher frequency terms of a trigonometric basis. Therefore, in practice we must resort to cleverer approximating formulations. In particular splines are well suited for fine-grained approximation while maintaining a low degree [26]. For our purposes we require a basis for the splines, differing from the standard treatment of approximating splines, which is mainly concerned with cubic splines.

**Example 5.** *Figure 1 demonstrates one of the disadvantages of using a polynomial basis – polynomials blow up at the edges. Because of this, approximations become increasingly inadequate towards the edges. Furthermore, this poses significant problems when using a polynomial fit in a pipeline of algorithms, wherein errors can propagate and multiply.*

### B-Splines

Usually splines are introduced as a way to interpolate discrete points without resorting to an inordinately high degree polynomial. It turns out that splines are also excellent at constructing least squares approximating functions,



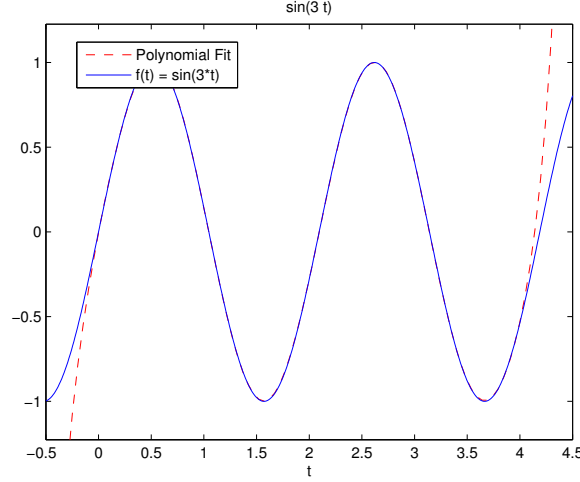


FIGURE 1. Blow up for a polynomial fit

but for that we require a spline basis. B-Splines are a special kind of spline which can be constructed through a basis [26].

**Definition 3.1.** *Given knot vector  $a = t_0 < t_1 < t_2 < \dots < t_{m-1} < t_m = b$ , a function  $\tilde{f}$  is a spline of degree  $n$  if:*

1. *On each interval  $(t_i, t_{i+1})$ ,  $\tilde{f}$  is a polynomial of degree  $n$ .*
2.  *$\tilde{f}$  is continuous and twice differentiable everywhere, including the knot points,  $\{t_i\}$  [26].*

**Definition 3.2.** (B-Spline) *Given a knot vector  $\mathbf{T} = \{t_0, t_1, \dots, t_m\}$ , where  $\mathbf{T}$  is nondecreasing, recursively define the basis functions as*

$$N_{i,0}(t) = \begin{cases} 1 & \text{if } t_i \leq t < t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,j}(t) = \frac{t - t_i}{t_{i+j} - t_i} N_{i,j-1}(t) + \frac{t_{i+j+1} - t}{t_{i+j+1} - t_{i+1}} N_{i+1,j-1}(t)$$

Where  $j$  is the degree of the basis, and for a given  $j$  there are  $m - j$  elements.

Then

$$\tilde{f}(t) = \sum_{i=0}^{m-j-1} \alpha_i N_{i,j}(t)$$

is a B-Spline of degree  $j$  [26].

B-splines are highly localized and approximate a Gaussian, as their degree increases. In a given family, there are more splines at lower degree. At higher degrees, the splines lack influence near the edges of the knot vector. We illustrate some B-spline bases elements in Figure 2.

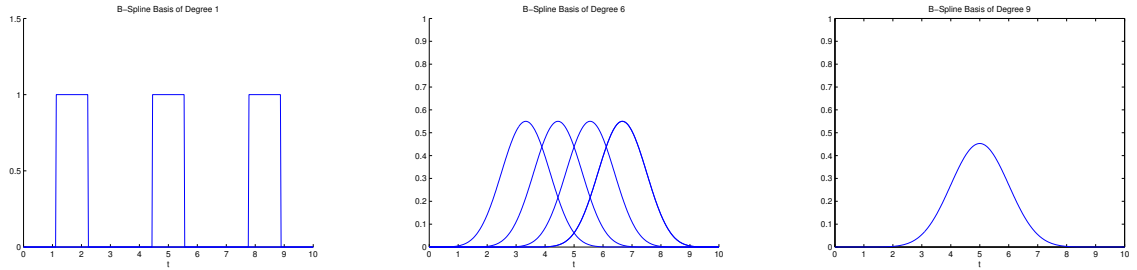


FIGURE 2. Illustration of a few different B-Spline bases elements

Notice how there are fewer higher degree basis elements, and that their influence doesn't much extend to the edges. This is actually of practical concern, while polynomial approximations tend to blow up at the edges, B-spline approximations die down. To combat this effect, we employ phantom knots. Phantom knots are additional knots that are artificially added to the edges of the knot vector. For instance, the knot vector  $\mathbf{T} = \{0, 1, 2, 3, 4, 5\}$  could be augmented to include phantom knots at  $-1$  and  $6$ , so that the new knot vector would be  $\mathbf{T}^* = \{-1, 0, 1, 2, 3, 4, 5, 6\}$ .

**Example 6.** (Phantom Knots) *In this example we fit the function*

*$f(t) = \sin(t) + \cos(t) + 0.3t$  using a B-spline basis on the interval  $[-10, 10]$ . The fits have been shifted vertically to aid visual comparison.*

Example 6 makes it clear that phantom knots drastically improve the

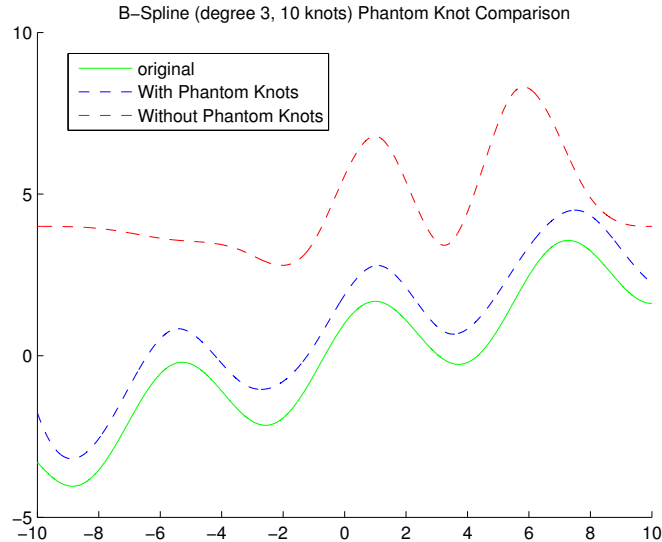


FIGURE 3. The effect of the inclusion of phantom knots

fidelity of B-spline fits; and they are indeed an integral part of an effective B-spline regression. There is yet another technique which can be applied to least squares fitting – the inclusion of a smoothing parameter [28, 27].

#### Smoothing Splines: Adding a Penalty Term

Often times in interpolation, the data we seek to model may contain wild oscillations. These fluctuations may be undue artifacts from the collection procedure, random noise and glitches, or even inherent to the data-generating system itself. In any case, attempting to fit a model to such overly-complex data can lead to very poor generalization results. As such, controlling for the complexity of the interpolant occasionally is desirable, even if at the cost of increased residual error. Similar considerations apply to the least squares differences problem.

We define the complexity of a fit  $\tilde{f}$  to be its total roughness, i.e.  $\int_{\Omega} \tilde{f}''^2 d\mu$ . From this we write the new regularized optimization objective, with  $\lambda \geq 0$  an

arbitrary regularization constant [28, 27]:

$$E = ||\mathbf{F} - \mathbf{\Phi}\alpha||^2 + \lambda \int_{\Omega} \left( \sum_i \alpha_i \phi_i'' \right)^2 d\mu$$

The former term is the familiar least squares error, while the latter term is new and seems formidable. However, some calculation shows that this penalty term can be written simply in matrix form.

$$\sum_i \alpha_i \phi_i'' = \begin{bmatrix} \phi_1'' & \phi_2'' & \cdots & \phi_n'' \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix}$$

So that

$$\left( \sum_i \alpha_i \phi_i'' \right)^2 = \alpha^T \begin{bmatrix} \phi_1''\phi_1'' & \phi_1''\phi_2'' & \cdots & \phi_1''\phi_n'' \\ \phi_2''\phi_1'' & \phi_2''\phi_2'' & \cdots & \phi_2''\phi_n'' \\ \vdots & \vdots & \ddots & \vdots \\ \phi_n''\phi_1'' & \phi_n''\phi_2'' & \cdots & \phi_n''\phi_n'' \end{bmatrix} \alpha$$

and

$$E = ||\mathbf{F} - \mathbf{\Phi}\alpha||^2 + \lambda \alpha^T \mathbf{M} \alpha$$

where

$$M = \begin{bmatrix} \int \phi_1''\phi_1'' & \int \phi_1''\phi_2'' & \cdots & \int \phi_1''\phi_n'' \\ \int \phi_2''\phi_1'' & \int \phi_2''\phi_2'' & \cdots & \int \phi_2''\phi_n'' \\ \vdots & \vdots & \ddots & \vdots \\ \int \phi_n''\phi_1'' & \int \phi_n''\phi_2'' & \cdots & \int \phi_n''\phi_n'' \end{bmatrix}$$

is a symmetric matrix.

Notice how this problem is still convex, since  $\mathbf{M}$  is positive semidefinite.

Therefore we expect to find a global minimum. Cast in this form, the regularized

least squares problem has a solution much the same as before:

$$\alpha = (\Phi^T \Phi + \lambda \mathbf{M})^{-1} \Phi^T \mathbf{F}$$

Furthermore, if we assume  $\mathbf{M}$  is positive definite, then  $\Phi^T \Phi + \lambda \mathbf{M}$  is invertible for  $\lambda > 0$ , so that the regularization parameter also enforces a global optimality to the solution of the least squares problem.

**Theorem 3.3.** *Let  $\mathbf{M}$  be positive definite and  $\lambda > 0$ . Then  $\Phi^T \Phi + \lambda \mathbf{M}$  is nonsingular [32].*

*Proof.* Let  $x \neq 0$ , and define  $y = (\Phi^T \Phi + \lambda \mathbf{M})x$ . Then  $x^T y > 0$  since  $\Phi^T \Phi$  is positive semidefinite and  $\mathbf{M}$  is positive definite:

$$x^T y = x^T \Phi^T \Phi x + \lambda x^T \mathbf{M} x > 0$$

Therefore  $y \neq 0$ , and since  $x$  was arbitrary we conclude  $\Phi^T \Phi + \lambda \mathbf{M}$  is nonsingular. □

Theorem 3.3 ensures that we can invert  $\Phi^T \Phi + \lambda \mathbf{M}$  as required by the normal equation solution for the least squares problem. However, as a matter of practice, one might find the presence of the ill-conditioned matrix  $\Phi^T \Phi$  unfortunate and seek to ensure numerical stability by choosing  $\lambda$  large enough.

It is worth noting the analysis above did not exploit any property of B-splines, and as such it applies to any basis functions. Moreover, as  $\lambda \rightarrow \infty$  the fit approaches the best-fit line. In that sense, the regularization imposed by our smoothing parameter is one that seeks to enforce as little curvature as possible. By varying  $\lambda$  we can choose the trade-off between data fidelity and smoothness of fit.

**Example 7.** (Regularized B-spline Regression) *Example fits for different regularization values (Figure 4). Notice how high regularization flattens the fit. In the limit, a regularized B-spline regression becomes the best fit line. The curves have been shifted vertically to aid visual comparison.*

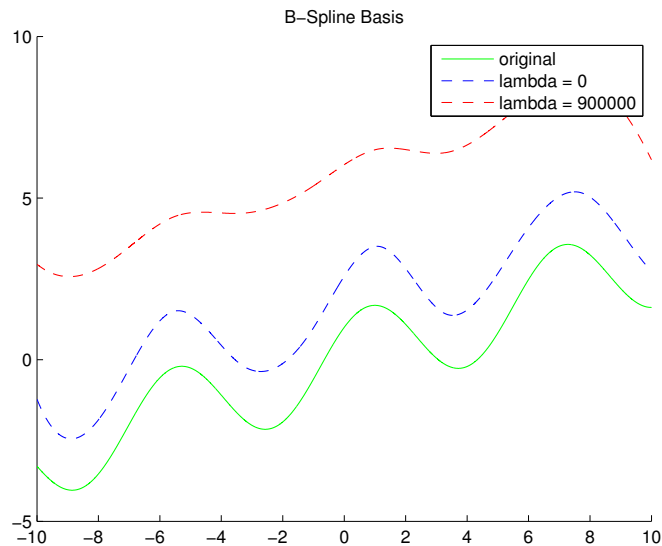


FIGURE 4. The effect of regularization on B-Spline Regression

### Efficiently Evaluating B-Splines

Recall from Definition 3.2 that B-splines are recursively defined in terms of lower degree B-splines. This is problematic, since function evaluation of a basis element is exponential in its degree. In short, the naive recursive implementation of B-splines is wholly unsatisfactory. However, if the points of evaluation are known a priori, we can restrict our attention to a discrete mesh and build the basis functions bottom-up through dynamic programming. This algorithm runs in linear time with degree, and is described by Algorithm 3.

Table 1 details the time savings afforded by the bottom-up approach. Run was performed on a 1.7 Ghz Macbook Air with 4 Gb of memory running MATLAB 2010. B-splines of various degrees were evaluated on a 100-point grid. Notice how the bottom-up method is significantly more performant, in addition to scaling roughly linearly, unlike the naive method, which takes an order of magnitude more time to evaluate a degree 15 spline.

---

**Algorithm 3** B-spline Evaluation [6]

---

1. For  $i = 0, 1, \dots, m$

Initialize each  $N_{i,0}$  on discrete mesh as an array, where each element is the basis function evaluated at a mesh point.

2. For  $j = 2, 3, \dots, n$

Set  $N_{i,j}(t) = \frac{t-t_i}{t_{i+j}-t_i}N_{i,j-1}(t) + \frac{t_{i+j+1}-t}{t_{i+j+1}-t_{i+1}}N_{i+1,j-1}(t)$  using element-wise array operations. Each  $t$  here is one of the discrete mesh points.

---

### Summary

B-splines are a good tool for interpolation and regression because of their robust numerical properties. In this chapter we motivated the need for bases other than the trigonometric and polynomial, as well as develop a framework for regularizing the least squares solution by means of a roughness penalty. Finally we showed how B-splines can be efficiently evaluated by using a bottom-up approach. All these concepts are of paramount importance to our work with AFMs. The excellent properties of B-splines allow us to accurately apply least squares differences to AFM signals of varying time scales, while maintaining real-time computational performance via the bottom-up evaluation approach. Additionally, the free regularization parameter can be tuned to experiment-specific settings, leading to a more customizable least squares differences fit.

TABLE 1. Computational Complexity of B-Spline Evaluation (seconds)

Degree	Naive Method	Bottom-Up Method
1	0.0425	0.0228
2	0.0427	0.026
3	0.0428	0.0321
4	0.0431	0.0373
5	0.0436	0.0426
6	0.0439	0.048
7	0.0472	0.0534
8	0.0528	0.0468
9	0.0642	0.0629
10	0.0854	0.0694
11	0.129	0.066
12	0.2151	0.0797
13	0.3894	0.0853
14	0.7068	0.0686
15	1.4248	0.0927



## CHAPTER 4

### INPAINTING

Inpainting is the process of recovering missing information in images. There are many different inpainting algorithms, such as H1, TV-L1, LCIS, and many more [8, 33, 4]. However, they all operate roughly in the same way: the extrapolation of missing detail from known data.

Currently, the most popular paradigm for image inpainting is the variational approach, whereby an energy functional is minimized by evolving a partial differential equation [9, 3]. The state-of-the-art techniques exhibit a host of interesting and desirable properties, such as the ability to complete edges and extrapolate patterns in a nonlocal fashion [5]. Many papers have been written on the subject of variational inpainting, and we will not belabor the point. Instead in this chapter we aim to introduce a novel inpainting algorithm based on the successful submissions to the Netflix Prize [18]. While we have had great success applying standard inpainting algorithms to our problem domain, we view that the alternative algorithm proposed herein holds much promise: it has the dual capacity of pattern completion and propagation of irregular information.

As we shall see in the next chapter, this kind of image reconstruction plays an important role in our proposal for fast atomic force microscopy. By scanning more sparsely, we can achieve much quicker scans while reproducing accurate images by inpainting the unknown regions.

#### Netflix Prize

In 2009, Netflix awarded the Netflix Prize. The Netflix Challenge was an

open competition seeking the best collaborative filtering algorithm for predicting users' movie ratings, based on their previous predilections. All successful algorithms had one thing in common. At heart, they performed a kind of singular value decomposition to find a succinct description of each users' movie ratings:

Formally, if  $\mathbf{B}$  is the matrix of movie ratings, with users as rows and movies as columns, then the winning algorithm sought to write  $\mathbf{B}$  in a simpler form.

$$\mathbf{B} \approx \mathbf{P}\mathbf{V}$$

$$\begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1l} \\ b_{21} & b_{22} & \cdots & b_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nl} \end{bmatrix}_{n \times l} \approx \begin{bmatrix} p_{11} & \cdots & p_{1k} \\ p_{21} & \cdots & p_{2k} \\ \vdots & \ddots & \vdots \\ p_{n1} & \cdots & p_{nk} \end{bmatrix}_{n \times k} \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1l} \\ \vdots & \vdots & \ddots & \vdots \\ v_{k1} & v_{k2} & \cdots & v_{kl} \end{bmatrix}_{k \times l}$$

This decomposition has an illuminating interpretation. Each user is associated with a row  $P_i$  in  $\mathbf{P}$  and each movie is associated with a column  $V_j$  in  $\mathbf{V}$ . Then the rating  $b_{ij}$  corresponds to the inner product between  $P_i$  and  $V_j$ . In essence we are measuring how well a user's preferences match with a movie's descriptors. The number of salient features  $k$  used for matching is arbitrarily chosen and constitutes a dimensionality reduction.

In general we cannot hope to reconstitute  $\mathbf{B}$  exactly if  $k$  is less than the number of nonzero singular values of  $\mathbf{B}$ , but we can get close. In fact, singular value decomposition provides a way to factor an arbitrary matrix into the product of smaller, simpler matrix. However, such expediency is not possible in the Netflix Prize because the matrix  $\mathbf{B}$  has unknown entries. Instead, standard gradient descent provides a simple method to estimate  $\mathbf{P}$  and  $\mathbf{V}$  [18].

Let  $\tilde{\mathbf{B}} = \mathbf{P}\mathbf{V}$  be an approximation to  $\mathbf{B}$ . Then define the reconstitution error by  $E = \|\mathbf{B} - \mathbf{P}\mathbf{V}\|_F^2$ , the sum of the squares of all the entries of the matrix  $\mathbf{B} - \mathbf{P}\mathbf{V}$ . Then

$$E = \sum_i \sum_j (b_{ij} - \tilde{b}_{ij})^2$$

$$\frac{\partial E}{\partial P_q} = -2 \sum_i \sum_j (b_{ij} - \tilde{b}_{ij}) \frac{\partial \tilde{b}_{ij}}{\partial P_q} = -2 \sum_j (b_{qj} - \tilde{b}_{qj}) \frac{\partial \tilde{b}_{qj}}{\partial P_q}$$

Since  $\tilde{b}_{ij}$  does not depend on  $P_q$  unless  $i = q$ . Therefore, we can write

$$\frac{\partial E}{\partial P_q} = -2 \sum_j (b_{qj} - \tilde{b}_{qj}) V_j$$

Gathering the terms, we can express all derivatives simultaneously by

$$\frac{\partial E}{\partial \mathbf{P}} = -2(\mathbf{B} - \mathbf{P}\mathbf{V})\mathbf{V}^T$$

Similarly, we can derive the partial derivative with respect to  $\mathbf{V}$ :

$$\frac{\partial E}{\partial \mathbf{V}} = -2\mathbf{P}^T(\mathbf{B} - \mathbf{P}\mathbf{V})$$

Armed with these derivatives, we can perform simultaneous dual gradient descent on the nonlinear optimization problem. We start with a random guess for  $\mathbf{P}$  and  $\mathbf{V}$  and follow the gradient (Algorithm 4). It is important to know that the minimum found will be local in nature, and in fact, notice that if  $\mathbf{P}$  and  $\mathbf{V}$  are both initialized to 0, then gradient descent will be at a standstill. Moreover, the algorithm may fail to converge if the learning rate is set too high, or we pick a particularly bad starting position. Nonetheless, this dual gradient descent algorithm is powerful because of its strong convergence properties [18, 30].

For some positive learning rate  $\tau$  and a mask  $\mathbf{M}$  which is 0 where  $\mathbf{B}$  is unknown and 1 otherwise, the update rule is

$$\mathbf{P}' = \mathbf{P} + \tau[(\mathbf{B} - \mathbf{P}\mathbf{V}) \circledast \mathbf{M}]\mathbf{V}^T$$

$$\mathbf{V}' = \mathbf{V} + \tau \mathbf{P}^T[(\mathbf{B} - \mathbf{P}\mathbf{V}) \circledast \mathbf{M}]$$

where  $\circledast$  indicates element-wise multiplication. The choice of  $\tau$  is application dependent and a bit of a black-art. Too small a  $\tau$  leads to slow convergence, whereas if  $\tau$  is too large the algorithm may altogether diverge.

**Example 8.** Texture completion

*Example of the Netflix Algorithm's capacity to complete patterns. An artificial binary matrix exhibiting a checkerboard pattern can be recovered completely despite damage (Figures 5 and 6). Since the checkerboard is a highly regular pattern, the Netflix Algorithm finds a simple common basis with which to succinctly describe the rows of the matrix. Experiment was performed on 1.7 Ghz i5 processor Macbook Air with 4 Gb of memory and MATLAB 2010. While in the limit complete recovery is achieved (Figure 5), it does take time. Table 2 shows the, admittedly slow, convergence for the first few iterations.*

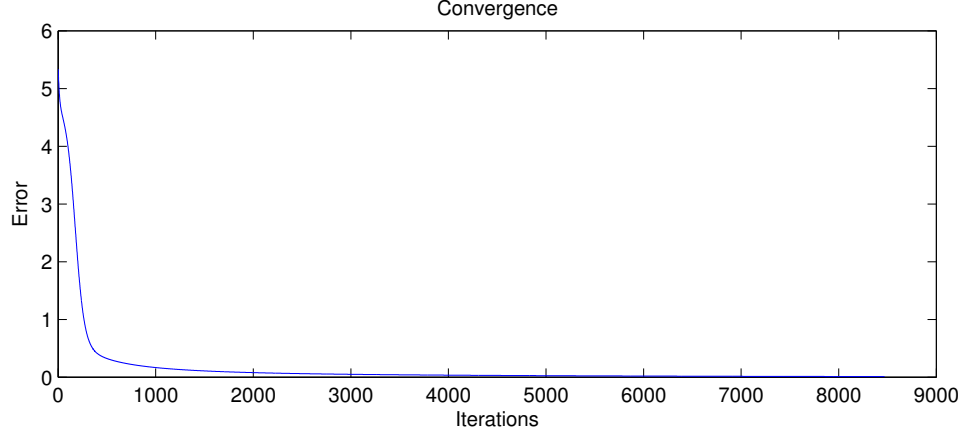


FIGURE 5. Convergence of error after many iterations

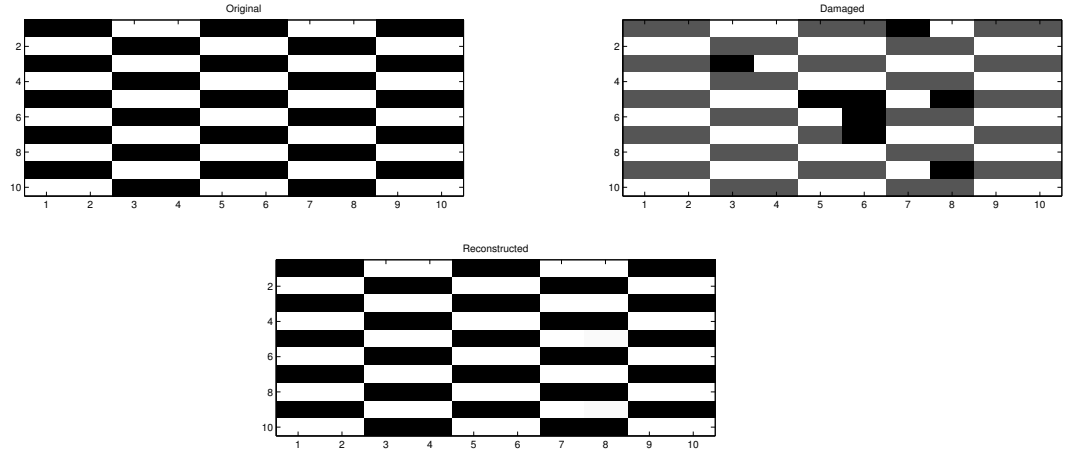


FIGURE 6. The Netflix Algorithm repairing a damaged pattern

---

**Algorithm 4** Netflix Algorithm

---

1. Randomly initialize the entries of  $\mathbf{P}$  and  $\mathbf{V}$

2. While convergence criterion not met

$$\text{Set } \mathbf{P} = \mathbf{P} + \tau[(\mathbf{B} - \mathbf{P}\mathbf{V}) \circledast \mathbf{M}]\mathbf{V}^T$$

$$\text{Set } \mathbf{V} = \mathbf{V} + \tau\mathbf{P}^T[(\mathbf{B} - \mathbf{P}\mathbf{V}) \circledast \mathbf{M}]$$


---

TABLE 2. Errors for the first twenty iterations of the Netflix Algorithm

Iteration	Error
0	9.8233
1	9.4105
2	9.0556
3	8.7489
4	8.4828
5	8.251
6	8.0481
7	7.8701
8	7.7131
9	7.5743
10	7.451
11	7.3412
12	7.2429
13	7.1547
14	7.0752
15	7.0033
16	6.9379
17	6.8783
18	6.8236
19	6.7733
$\vdots$	$\vdots$

## Penalized Dictionary Inpainting

As illustrated by Example 8, the method of finding a low dimensional decomposition of a matrix with missing data can be useful in recovering damaged patterns. With our proposed penalized dictionary inpainting we hope to leverage that power but in an image processing setting. In effect, our aim is to create a dictionary of image patches which collectively describe almost all the information in the image. Furthermore, by imposing a regularization penalty, we may enforce a desirable smoothing condition on the reconstructed image. For the case of images, this regularization favors the reasonable assumption of similarity between neighboring patches.

Before, we were concerned with reconstructing a low rank matrix from partial information. That is, we assumed the rows of the matrix shared a compact, common basis. To apply the Netflix Algorithm to images, we must instead collect all  $p \times p$  neighborhoods and express them as rows of the matrix  $\mathbf{B}$ . Concretely, for each square patch of the image we form a vector from serializing all the pixel values and put said vector into the neighborhood matrix  $\mathbf{B}$  (see Figure 7). Below we demonstrate this transformation (column concatenation) on an example pixel patch:

$$\begin{bmatrix} 14 & 250 & 253 & 255 \\ 13 & 14 & 250 & 254 \\ 10 & 12 & 14 & 250 \\ 10 & 10 & 12 & 13 \end{bmatrix} \Rightarrow \begin{bmatrix} 14 \\ 13 \\ 10 \\ 10 \\ 250 \\ 14 \\ 12 \\ 10 \\ 253 \\ 250 \\ 14 \\ 12 \\ 255 \\ 254 \\ 250 \\ 13 \end{bmatrix}$$

In this way, we avoid the inherent anisotropy of the Netflix Algorithm and also pick up nonlocal sensitivity – distant patches can be correlated [23]. Each row  $B_i$  of neighborhood matrix  $\mathbf{B}$  corresponds to some patch on the image. Since we expect the image to mostly feature smooth pixel transitions, it is reasonable that immediately adjacent patches be very similar. Hence, by regularizing we require that the preference vectors  $P_i$  of neighborhoods that differ by exactly one pixel shift be nearly the same:

$$||P_k - P_m||^2 < \epsilon$$

for adjacent patches  $k$  and  $m$ . Each neighborhood  $\mathcal{N}_p$  centered at pixel  $p$  may be



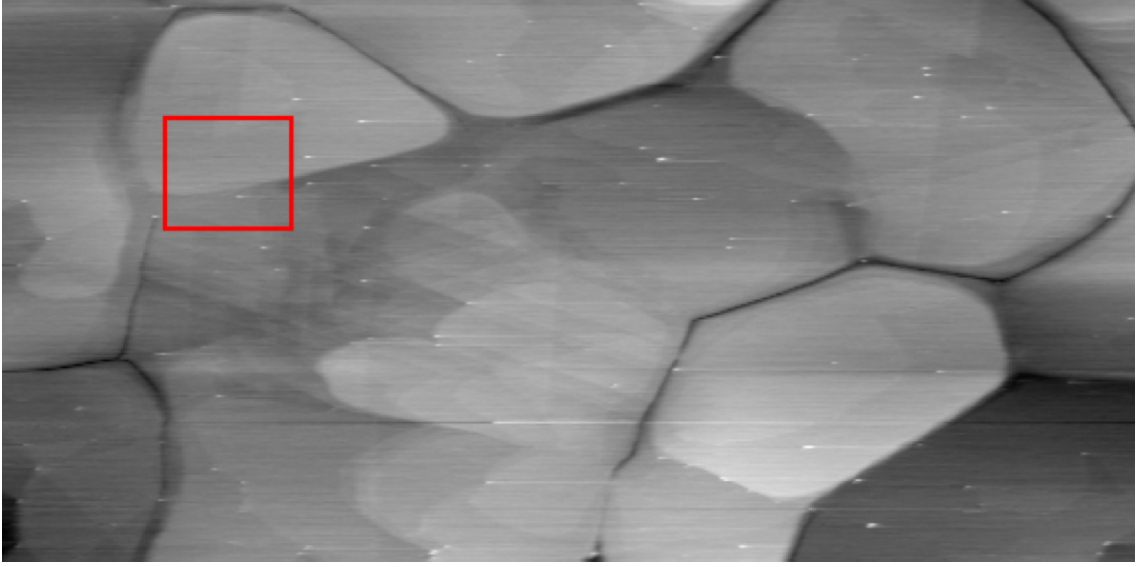


FIGURE 7. The box highlights a typical neighborhood of an image

adjacent to two, three, or four different neighborhoods, contingent on it being an edge case. Together, these are interpretable as a derivative penalty in the error formulation. For instance, if we let the neighborhood size be of a single pixel, then the penalty corresponds to the square of magnitude of the gradient at each pixel. To encode this in matrix form, recall that left matrix multiplication yields linear combinations of rows, so that each  $A_p^*$  picks out the difference for one direction. Furthermore, the magnitude of each of these differences can be written as a quadratic form. Hence, after taking derivatives we can promptly combine the terms into the penalty matrix  $\mathbf{M}$ .

$$E = \|\mathbf{B} - \mathbf{P}\mathbf{V}\|_F^2 + \lambda \sum_p \{ \|A_p^{(left)}\mathbf{P}\|^2 + \|A_p^{(right)}\mathbf{P}\|^2 + \|A_p^{(up)}\mathbf{P}\|^2 + \|A_p^{(down)}\mathbf{P}\|^2 \}$$

Where  $\lambda$  is a regularization parameter and

$$A_p^{left} = \begin{bmatrix} 0 & 0 & \cdots & 0 & -1 & 1 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

$$\begin{aligned}
A_p^{right} &= \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 & -1 & 0 & 0 & \cdots & 0 \end{bmatrix} \\
A_p^{up} &= \begin{bmatrix} 0 & \cdots & 0 & -1 & 0 & \cdots & 0 & 1 & 0 & \cdots \end{bmatrix} \\
A_p^{down} &= \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & -1 & 0 & \cdots \end{bmatrix}
\end{aligned}$$

with the 1 in the position corresponding to neighborhood  $\mathcal{N}_p$ 's row in  $\mathbf{P}$  and the  $-1$  in the position corresponding to the neighborhood above, below, and besides. For edge cases, we abide by the convention of an all 0s  $\mathbf{A}_p^*$  array for the missing direction.

Proceeding as before, we differentiate the error by  $\mathbf{P}$  and  $\mathbf{V}$ . Summing up the  $A_p^*$  we arrive at the derivatives

$$\begin{aligned}
\frac{\partial E}{\partial \mathbf{P}} &= -2(\mathbf{B} - \mathbf{P}\mathbf{V})\mathbf{V}^T + \lambda\mathbf{M}\mathbf{P} \\
\frac{\partial E}{\partial \mathbf{V}} &= -2\mathbf{P}^T(\mathbf{B} - \mathbf{P}\mathbf{V})
\end{aligned}$$

$\mathbf{M}$  is a tribanded matrix with entries in  $\{-1, 0, 1, 2, 3, 4\}$ , as displayed in Figure 8. The space between the bands and values are governed by the image dimensions: light blue corresponds to 0, dark blue to -1. Yellow, orange, and red correspond to 2, 3, and 4 respectively. Correctly forming the penalty matrix is challenging, we refer the reader to the attached source code for a more thorough demonstration. Briefly, the off-diagonals correspond to neighboring pixel-patches, hence the periodic edge cases. As with the Netflix Algorithm, we pursue a gradient descent strategy for finding the minimum, but with the additional complexity of dismantling the image into neighborhoods and rebuilding it from the computed solution (Algorithm 5).

### Summary

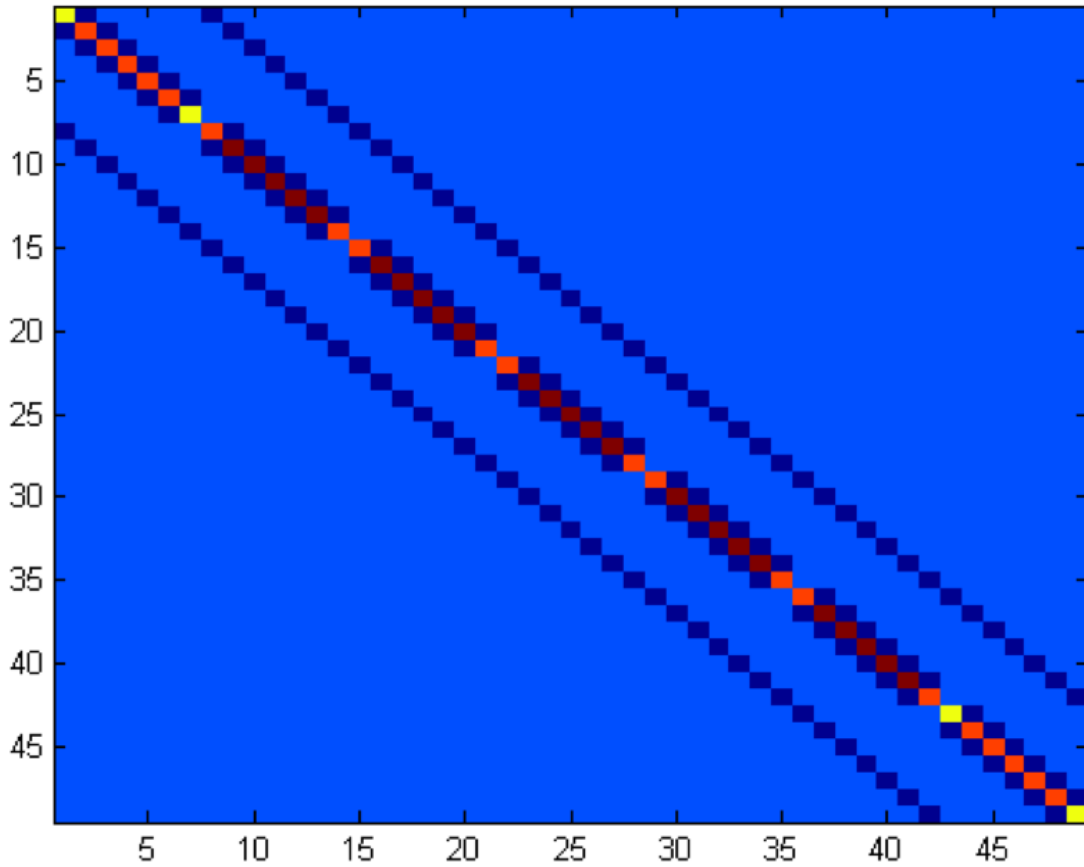


FIGURE 8. The repeating pattern corresponds to rows in the image

Penalized dictionary inpainting borrows heavily from the Netflix algorithm's pattern insights. However, by adding a penalization term we hope to both enforce geometric considerations upon the reconstructed image, as well as aid in the propagation of information. Mathematically the  $\lambda$  term plays a role similar to that of diffusion in the heat equation. To see this, consider a simplified error formulation with only the regularized penalty term. In this case, it is easy to see that it penalizes gradients. Figure 9 shows the dual capacity of the PDI algorithm, both pattern completion and gradient diffusion, as applied to a checkerboard

---

**Algorithm 5** Penalized Dictionary Inpainting

---

1. Form neighborhood matrix from the image by placing each neighborhood  $\mathcal{N}_p$  centered at pixel  $p$  as a row vector in  $\mathbf{B}$
  2. Randomly initialize the entries of  $\mathbf{P}$  and  $\mathbf{V}$
  3. While convergence criterion not met  
Set  $\mathbf{P} = \mathbf{P} + \tau \{[(\mathbf{B} - \mathbf{P}\mathbf{V}) \circledast \mathbf{M}]\mathbf{V}^T - \lambda\mathbf{M}\mathbf{P}\}$   
Set  $\mathbf{V} = \mathbf{V} + \tau\mathbf{P}^T[(\mathbf{B} - \mathbf{P}\mathbf{V}) \circledast \mathbf{M}]$
  4. Recombine neighborhoods into an image. Overlapping areas are taken as the average of all its respective values.
- 

image. With no regularization, a perfect reconstruction of the original pattern is recovered, courtesy of the low-rank completion properties of PDI. With higher regularization ( $\lambda = 6$ ) the algorithm starts favoring neighborhood similarity over completing patterns. This blurring ability is integral in propagating known information into unknown territory when the image exhibits irregularity, while PDI's grounding on linear algebraic structure favors maintaining and completing patterns. The run, taking 1.3 seconds, was performed on a 1.7 Ghz Macbook Air with 4 Gb of memory. Notice how higher regularization eschews reconstruction accuracy for gradient propagation (Table 3).

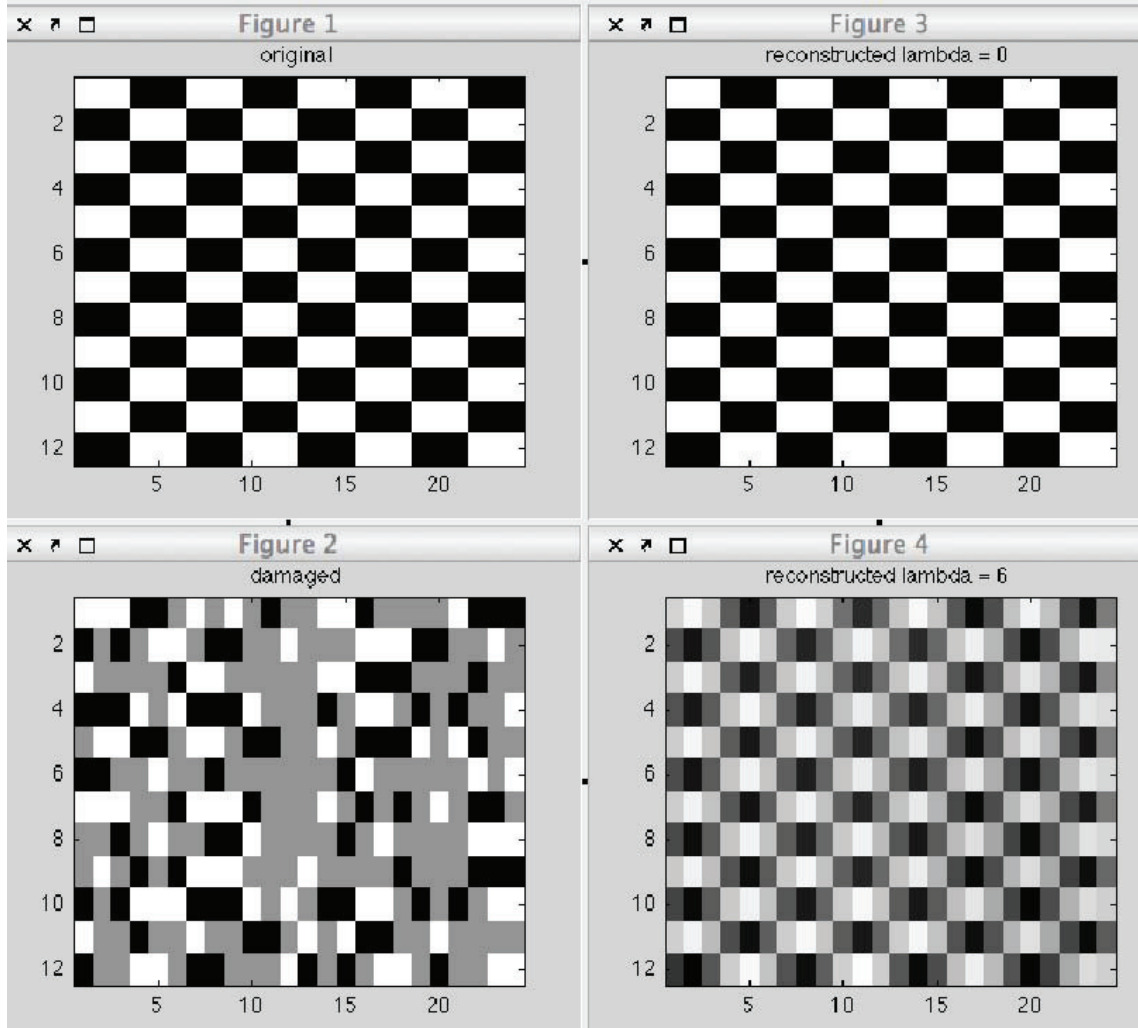


FIGURE 9. Penalized dictionary inpainting pattern completion with different  $\lambda$ s

TABLE 3. Convergence of PDI

Iteration	Error ( $\lambda = 0$ )	Error ( $\lambda = 6$ )
150	9.781237	21.179384
300	6.778566	16.812058
450	3.999007	12.472860
600	1.322371	10.441871
750	0.489060	9.479906
900	0.220346	8.943725

## CHAPTER 5

### EXPERIMENTAL RESULTS

Equipped with the theoretical developments of the previous chapters, we are now able to detail how to improve the speed of atomic force microscopy scans. However, first we give a brief background of AFMs proper. We introduce the common method of data processing currently in use and highlight its issues before presenting our own alternative – sparse cycloidal and spiral scans coupled with least squares differences noise removal and inpainting for image rendering.

#### Atomic Force Microscopy

Atomic force microscopy is a type of microscopy which acquires its images not by optical means, but by measuring the force interactions between a mechanical cantilever and sample. High resolution images can be obtained by methodically scanning an area of a sample and recording these minute interactions. Atomic force microscopes (AFMs) can attain sub-nanometer resolution and, under ideal circumstances, even achieve atomic resolution. In addition to their fantastic imaging capabilities, atomic force microscopes can also operate in the absence of a vacuum and even in liquid environments. Furthermore, samples need not be especially prepared for imaging, as with alternative microscopy approaches, such as tunneling or scanning electron microscopes. For these reasons, AFMs have garnered some prominence in applied scientific research [11].

The way atomic force microscopes work is surprisingly simple: A laser is shone on a reflective cantilever and the reflected light is captured by a photodiode. The cantilever has a needle-like tip which is deflected by interactions with the

surface of the sample. These deflections translate to changes in the photodiode’s measurements. Finally, to obtain an image, the entire sample table is moved under the tip by very precise piezoelectric motors. See Figure 10 for a schematized drawing.

While AFMs have excellent resolution and performance characteristics, they are not without problems. They tend to be considerably slower than their scanning electron microscope counterparts. In addition, thermal fluctuations cause the laser measurements to drift [31, 25, 15]. Lengthy scan times further exacerbate the drift problem, as it introduces significant distortion to images. Even more insidiously is the drift and creep present in the piezoelectric motors. The operation of an atomic force microscope requires very finely tuned movements in the order of nanometers or less. The piezos can indeed be very precise, but they suffer from highly nonlinear behavior and hysteresis [14, 16]. Because of this, scanning a predetermined path is not always so straight-forward. Finally, because of the sensitivity of the AFM apparatus, it is particularly susceptible to vibrations and mechanical resonance.

The present cutting edge research in atomic force microscopy is concerned with increasing scan speed, the goal being “video rate” scans, i.e. image reproduction at several Hz [29]. The primary focus has been on improving the hardware, but recently alternative scan modes have been suggested; some with the hopes of forestalling the mechanical resonance problem [2, 10, 22], while others aim to find and track only the salient features [12, 13, 20].

### Raster Scan And Line Flattening

Raster scanning is the traditional scan technique and still very much the norm. In raster scanning, the needle moves back and forth across the sample in a sawtooth pattern. Because of the sharp transition in velocity at the extremes of



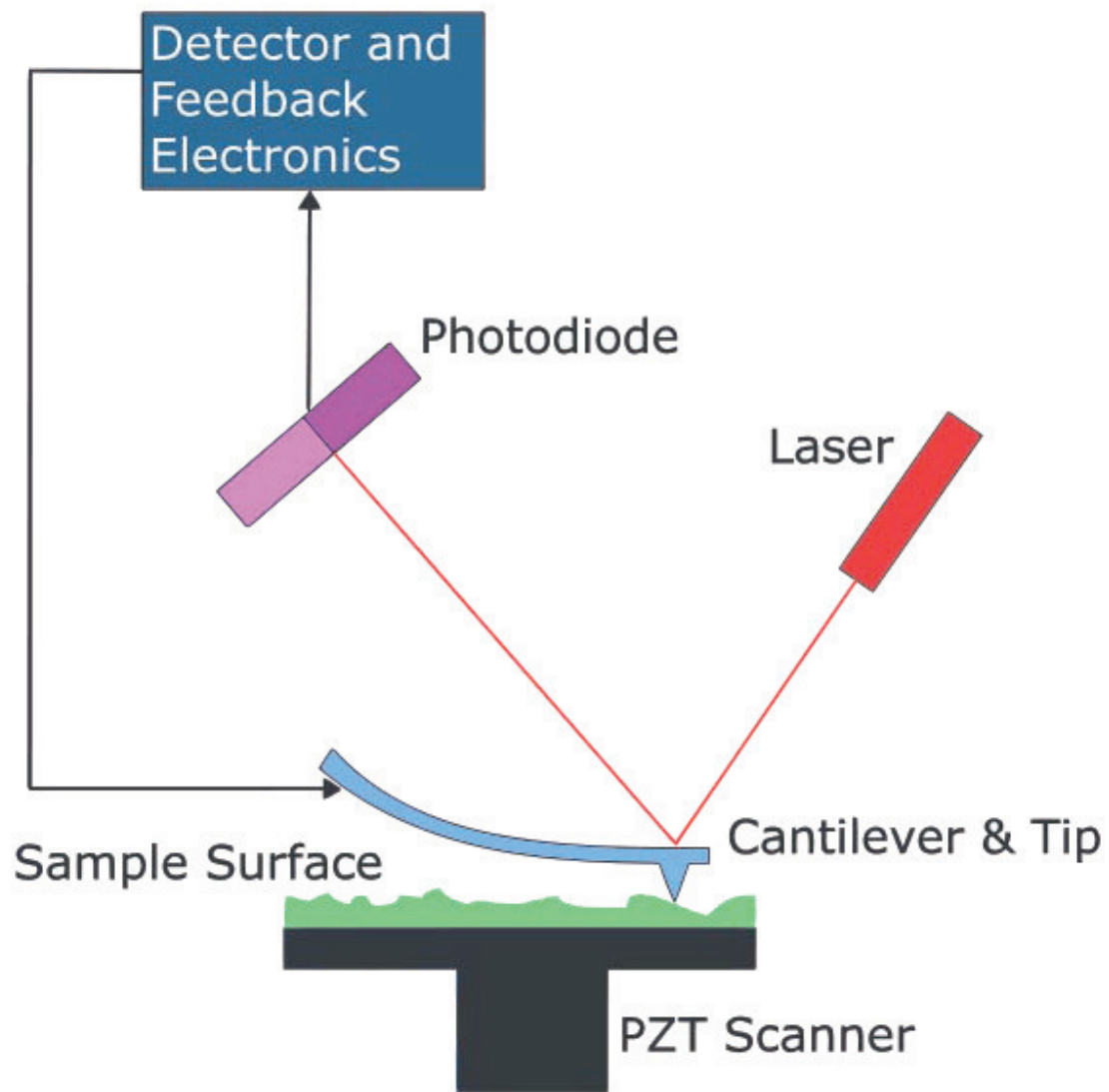


FIGURE 10. Atomic force microscopy schematic

[35]

the journey, mechanical vibrations are introduced to the machine setup. This presents problems when increasing the scan speed, as the higher frequencies can quickly cause resonance within the apparatus, thereby distorting the image.

An advantage of raster scanning is the simplicity of processing the raw image to remove thermal drift. By assuming that each scan line has an average intensity of zero and that the thermal drift is adequately modeled by a linear function, each line can be processed individually. This technique is called line flattening and it works quite well when the assumptions hold, albeit introducing some distortions. However, when the assumptions are faulty, significant information can be lost (Figure 12). Additionally, since each scan line is handled separately, there is no continuity enforced along the orthogonal direction, leading to visible striations. Nonetheless, raster scan with line flattening has remained the de facto standard in atomic force microscopy and does indeed perform admirably well in the vast majority of cases. Figure 11 depicts how line flattening is highly capable of removing thermal drift, which manifests itself as the gradient of the left image. While the recovered image on the right contains a lot of fine detail, it also exhibits the typical glitches and striations associated with line flattening.

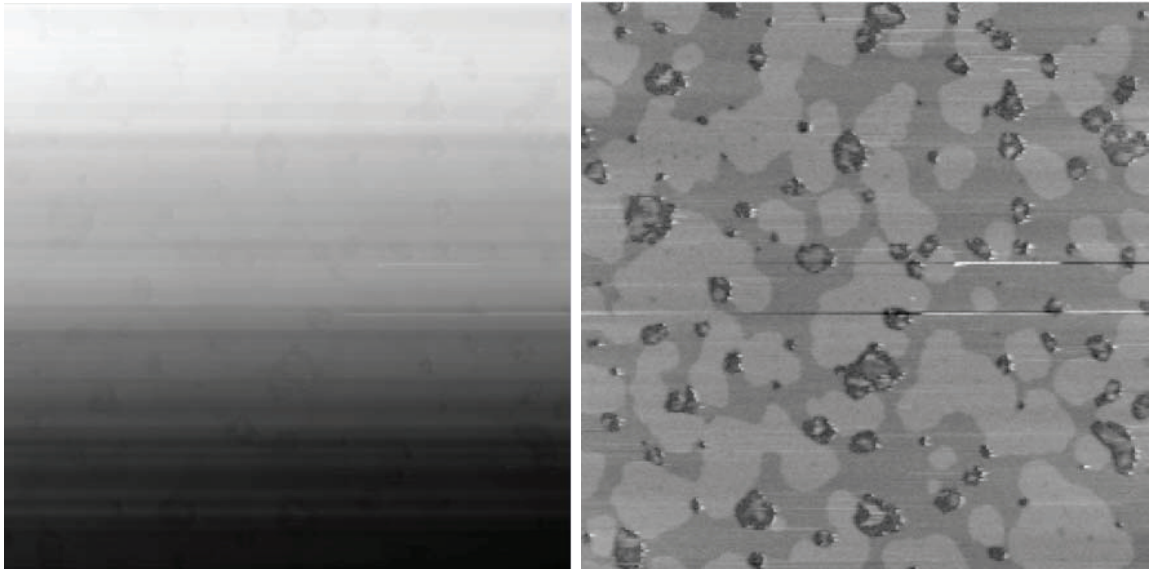


FIGURE 11. Line flattening of a raster scan with thermal drift

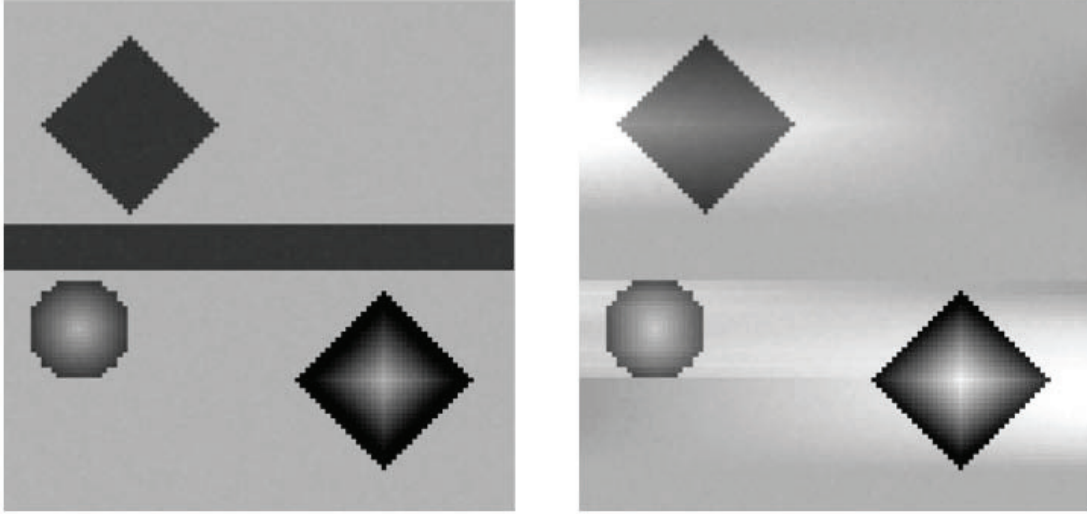


FIGURE 12. Distortions are introduced when assumptions do not hold

### Alternative Scan Paths

Recent literature has introduced alternative scan paths to address the mechanical resonance problem which has impeded increasing scanning frequency. Prominent among them are spiral and cycloid paths [21, 34]. Because these paths are comprised of arcs, they exhibit much more favorable frequency characteristics. To wit, cycloidal scans have only two frequencies in Fourier space whereas (constant linear velocity) spiral scans have bounded and small frequency when the region near the origin is excluded. By leveraging this property of the two scans, video rate and near video rate scanning has been demonstrated on regular AFMs without highly specialized equipment.

An intrinsic property of the cycloid scan is its self-intersections. To some degree, this redundancy indicates wastes of effort and time, but the increased tip velocity more than makes up for this. Moreover, these overlapping observations

allow us to use the powerful tools developed in Chapter 2 (least squares differences with B-splines) to remove thermal drift. So desirable is this property, that we propose the double cycloid scan to introduce intersections for the express purpose of employing the differences algorithm to thermal drift correction. Figures 13 and 14 show cycloid and double Archimedean spiral paths with their associated “bundle plots”. To understand these bundle plots, recall that any self-intersection in the scan path corresponds to a point on the sample substrate being visited twice. Each line segment in the bundle corresponds to one such self-intersection, connecting the two observations in time.

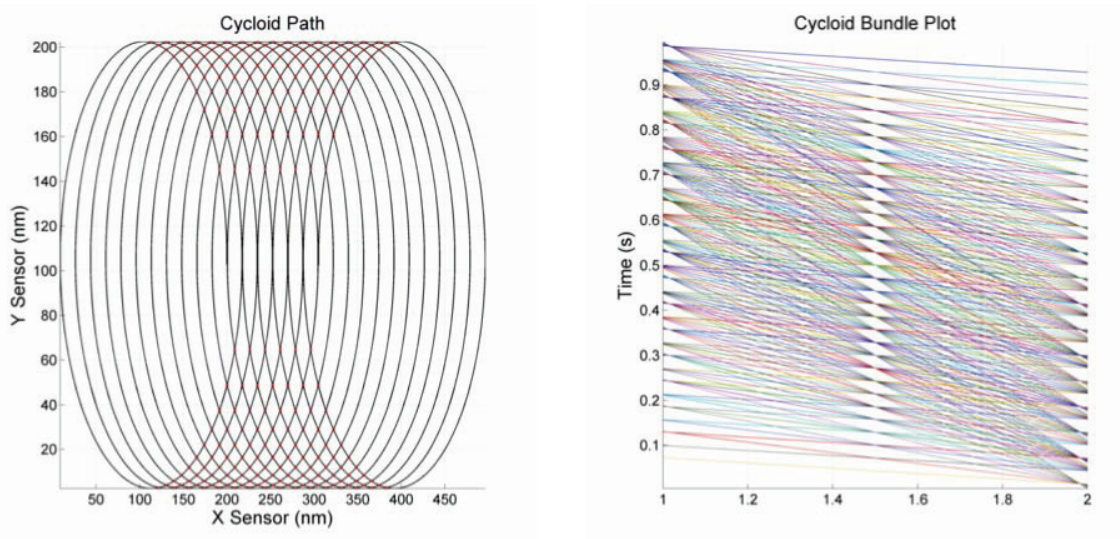


FIGURE 13. Cycloid scan path with its associated bundle plot

Before implementing these ideas on a physical atomic force microscope, we investigated the algorithms in a simulated environment. Producing drift corrected images from raw sensor data actually involves a complicated pipeline: To apply the differences algorithm to the collected data, we first identify all self-intersections by way of a quadtree – a data structure from computer science well-suited for such

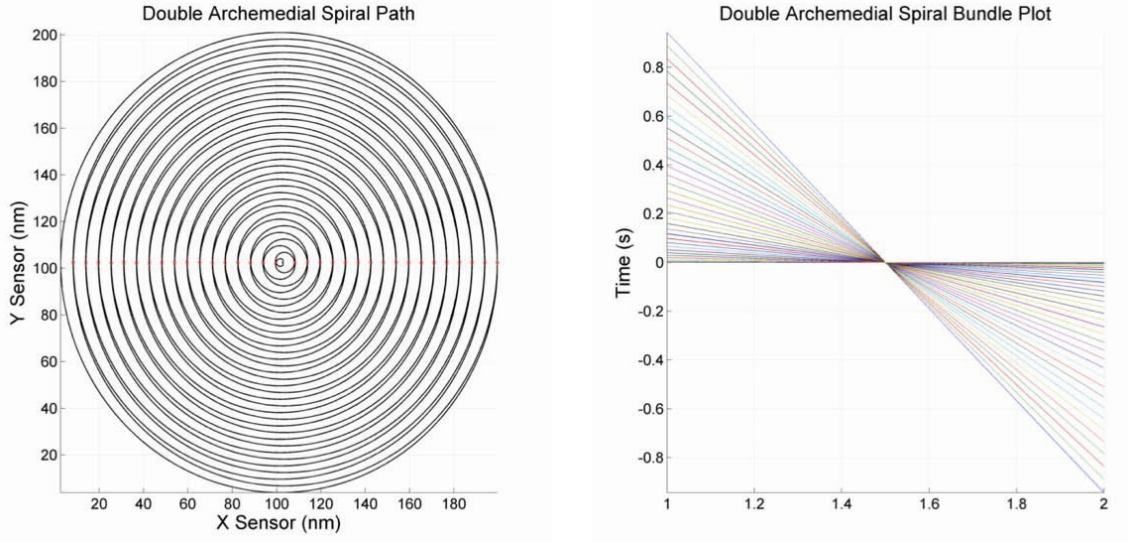


FIGURE 14. Spiral scan path with its associated bundle plot

tasks [17]. This process is in practice made more involved by the discrete nature of the AFM's signal. Only after thusly detrending the data, did we employ inpainting algorithms to fully render the image. Figures 15 and 16 demonstrate the dramatic effect background removal has on the rendering of cycloid scanned data. In particular, Figure 16 is almost completely devoid of any ringing or artifacts induced by the simulated thermal drift. These simulated results were confirmed on our experimental data, which displays similar improvement (Figure 20).

#### Inpainting And Sparsity

Inpainting plays a double role when it comes to alternative scans such as the double Archimedean spiral and cycloid scans. Firstly, by scanning more sparsely and inpainting the missing regions, we can capture images in less time. Secondly, by the undirected nature of these scans, it is unclear how to reconstitute an image from position and intensity triplets as compared to the simplicity of

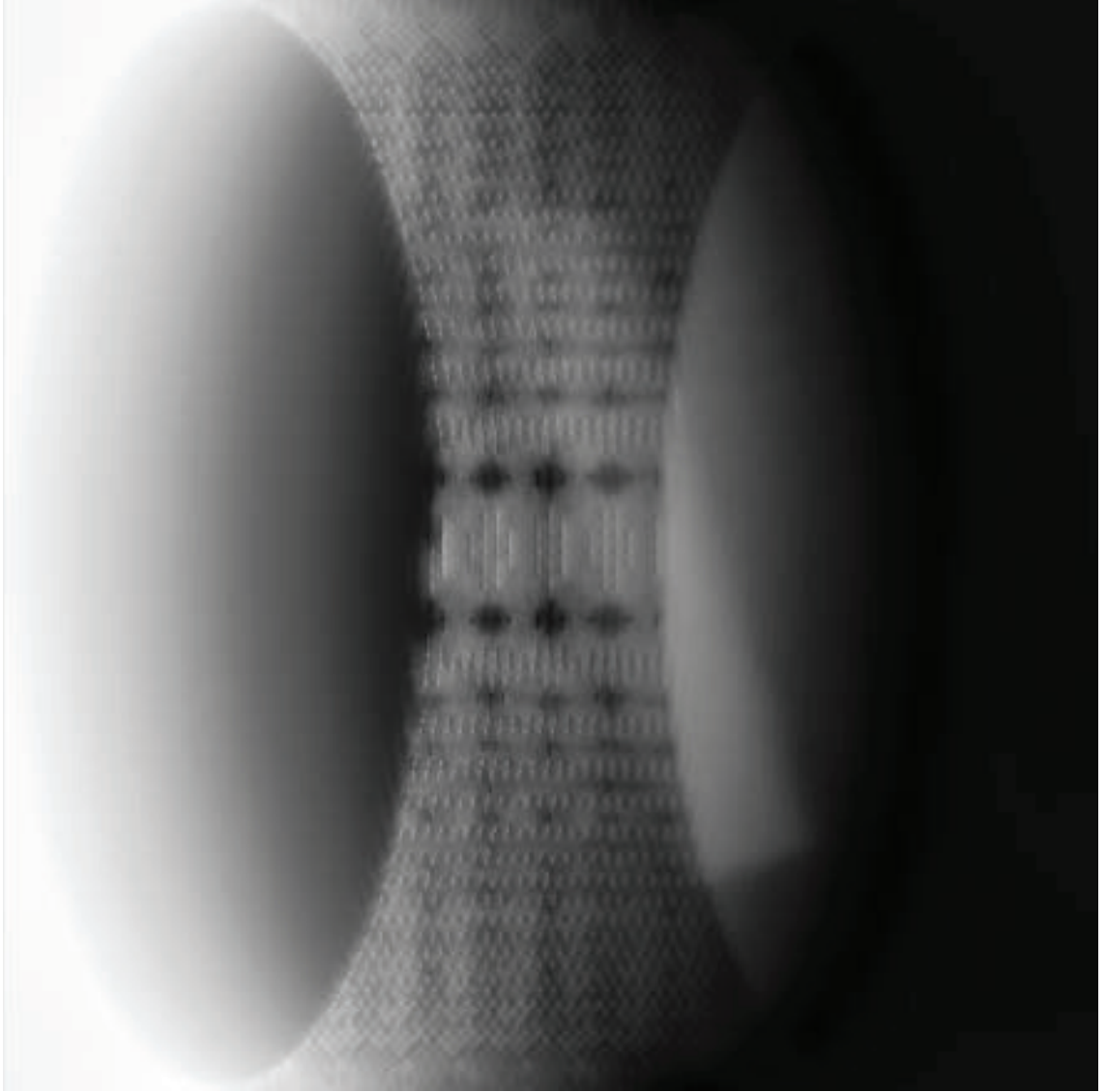


FIGURE 15. Image generated from a simulated cycloid scan sans drift correction

rendering raster images. The easy and correct solution is to inpaint the domain while holding the known data fixed [12].

While it had been our hope that Penalized Dictionary Inpainting (PDI) would prove to be a competitive algorithm for rendering AFM images, it is simply





FIGURE 16. Image generated from a simulated cycloid scan after drift correction

too computationally expensive at this time to run along side a video rate scanner. Such is also the case for more sophisticated inpainting algorithms like TV-L1, therefore in practice we used a fast implementation of the heat equation H1 algorithm for rendering our images [4]. Despite its computational shortcomings,

Penalized Dictionary Inpainting fared favorably against TV-L1 for the types of images we are interested in, when time is not a constraint. Figure 17 shows a side-by-side comparison between PDI and TV-L1. PDI successfully propagates information into the unknown regions and more closely resembles the original (Figure 7). TV-L1 on the other hand, makes for very crisp edges. Figure 18 is a screenshot of our algorithm as it converges to the optimal solution.

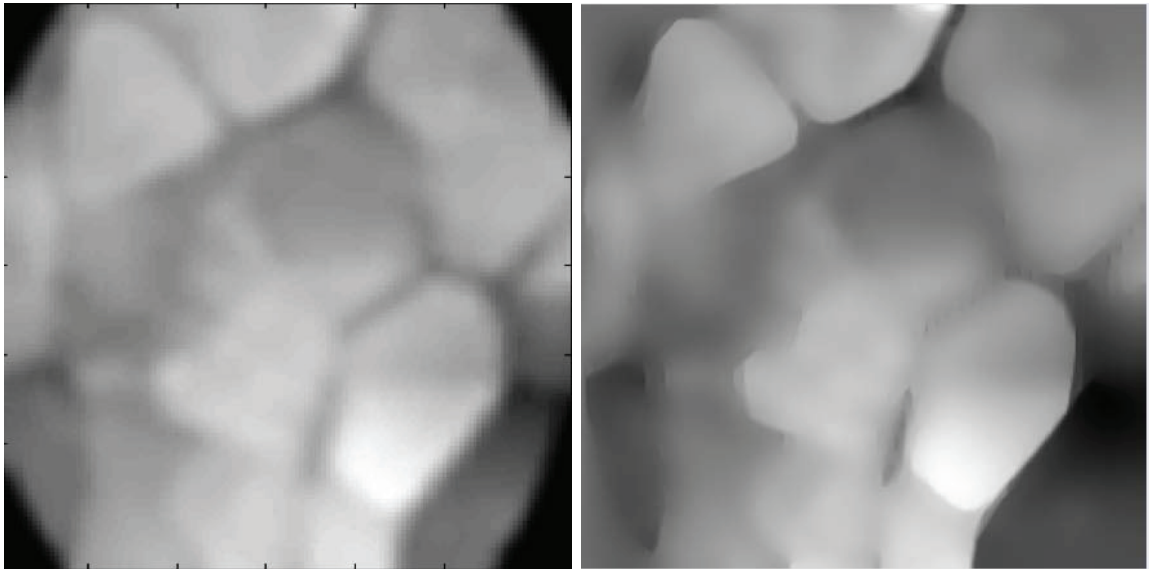


FIGURE 17. Comparison between PDI (left) and TV-L1 (right)

### Putting It All Together

We had access to an atomic force microscope at the Lawrence Berkeley National Laboratory which we programmed to use our routines for scanning. As detailed before, the Fourier spectra of cycloid and spiral scans allow for a much faster scan than possible with the conventional raster pattern. However, practical considerations limited our ability to scan as fast as we desired. Namely, the cantilever itself floats above the sample, kept in check by a z-axis piezo, whose latency and speed fundamentally limited how fast we could scan across high-relief



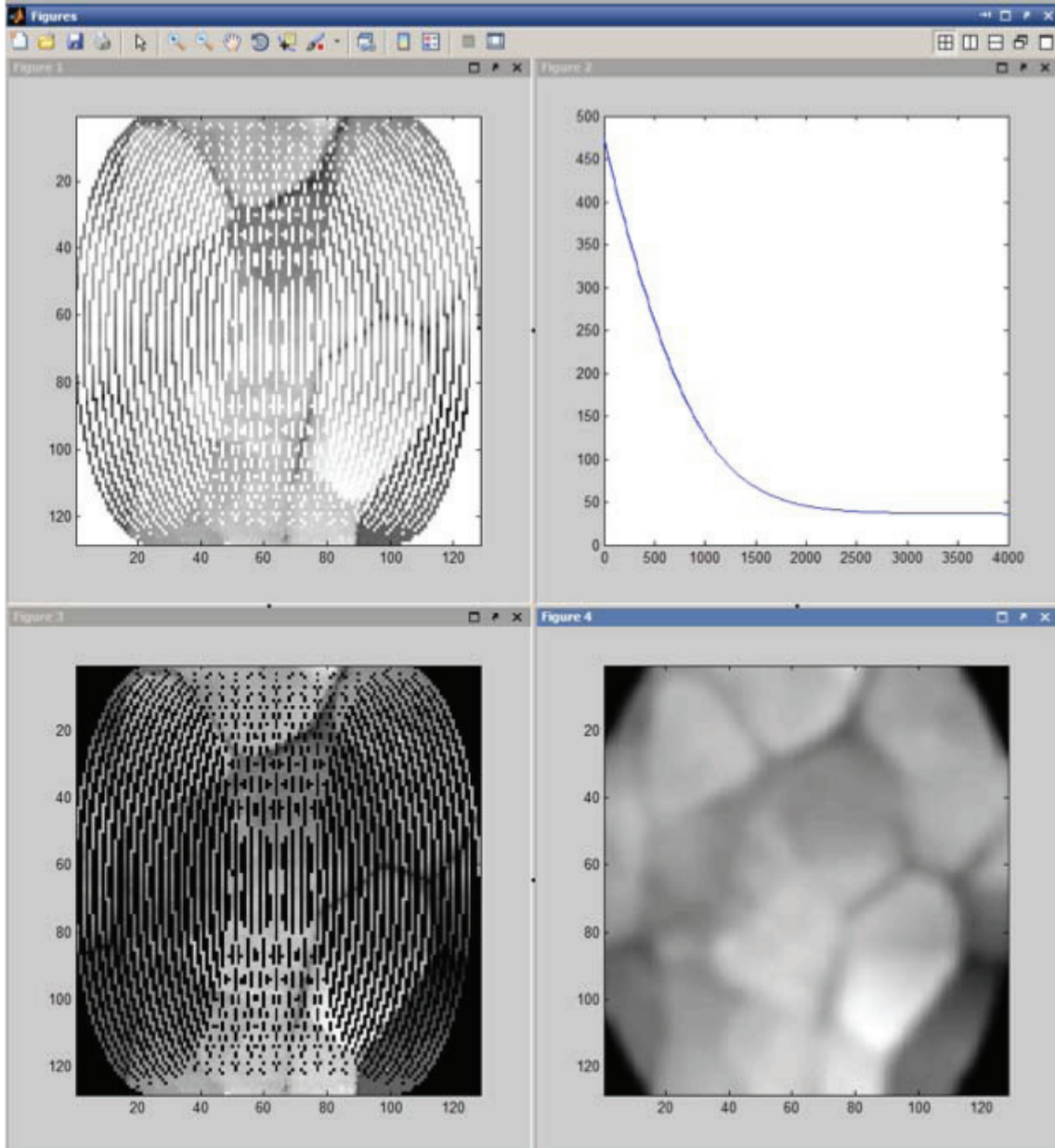


FIGURE 18. A screenshot demonstrating PDI's slow convergence

topography. Furthermore, the considerable nonlinearities in the piezoelectric motors driving the scan table made it difficult to localize the tip position. This problem is further compounded by the very low resolution position sensors that

were available to us. Despite these challenges, the images we gathered in the lab show much promise. Additionally, we did meet our goal of improving scan time while retaining image quality. In some cases we were able to increase the speed by four-fold. Figure 19 shows the successful imaging of two calibration grids using cycloid scan. The dense cycloid scan, albeit slow, yielded very clear images of the high relief topography. Figure 20 is a near real-time (0.25 Hz) rendering of spiral-scanned annealed gold with background noise removed via least square spline differences and H1 inpainting.

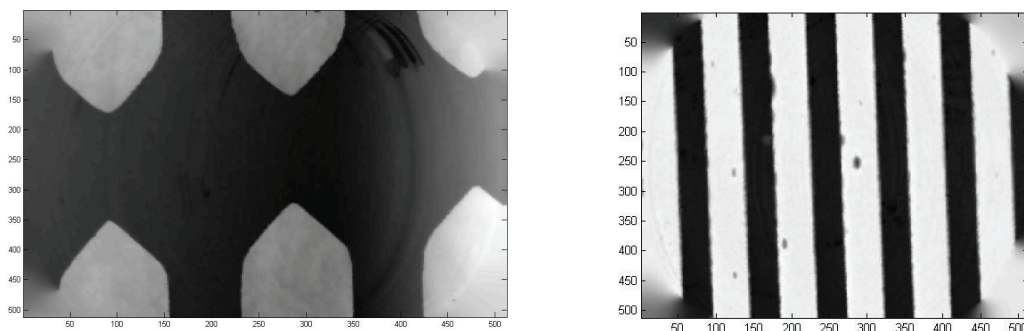


FIGURE 19. Cycloid scans of calibration grids

### Summary

In this chapter we showed how our methodology of non-raster scan paths, differences drift correction, and sparse image inpainting can be applied to atomic force microscopy to produce fast and accurate scans. Promising results were achieved when testing our techniques on an AFM in the Lawrence Berkeley National Laboratory.

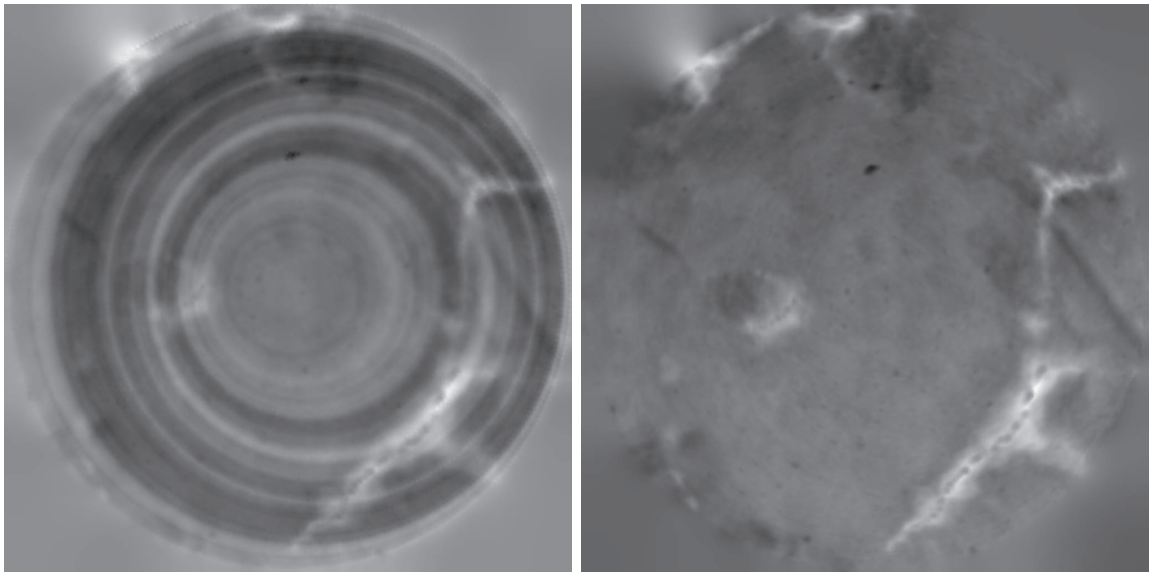


FIGURE 20. Before and after drift correction images of spiral scanned gold

## CHAPTER 6

### SUMMARY AND CONCLUSIONS

In this paper we presented two mathematical ideas for the analysis of data: the theory of differences and penalized dictionary inpainting. We also illustrated how they can be applied to the overarching idea of sparse, alternative scan paths for atomic force microscopy.

#### Least Squares Differences

The differences algorithm seems especially well-suited to the self-intersecting paths of fast atomic microscopy, as it provides a powerful alternative to the unsatisfactory line flattening approach. Least squares differences doesn't suffer from the internal inconsistencies and pathological distortions inherent to the established methods.

#### Penalized Dictionary Inpainting

Penalized dictionary inpainting exhibits non-local sensitivity, with its capability to complete regular patterns. However, it runs very slowly so it is not a competitive choice, given the recent innovations in solving L1-Regularized problems [19]. With increasing computational power, it may be that this algorithm becomes practical, especially because the matrix operations of the algorithm are highly parallelizable.

#### Future Work

There is still much to improve in applying differences to atomic force microscopy. Because of hardware limitations, the position sensors on an atomic force microscope do not have a very high precision. This shortcoming, coupled

with piezo creep and drift, lead to erroneous positional measurements. These can in turn spoil the fit generated by least squares differences. To combat this insidious problem, we have been investigating filtering the position sensor signal. In particular, in our future data collections, we would like to apply Kalman filtering, or perhaps the Viterbi algorithm, to better understand the path the atomic force microscope sampled. The primary difficulty in formulating these Markov models is characterizing the physics of the process evolution. Nonetheless, this information – albeit arcane – can be found within the literature [14, 16], hence we are confident significant strides can be made in this area.

SOURCE CODE

We present here source code for the relevant routines – `splinesmooth`, `bpoly`, `bspline`, and `pdi` – as well as two example scripts for how to use them. Additionally, `bsplineslow` is included for completeness and for comparison.

#### Listing 6.1. Differences Fitting Example Script

---

```

1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  % Least squares difference example script for how to use splinesmooth()%
3  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4
5  f = @(t) sin(t) + cos(t) + .3*t; % Simulated signal
6  T = linspace(-10,10,5000); % Discrete mesh
7  Ti = T(ceil(rand(2,400)*5000)); % Randomly generated temporal differences (intersection times)
8  Z = f(T);
9  Delta = f(Ti(2,:)) - f(Ti(1,:)); % Simulated observed differences
10
11  %%
12  figure; hold on;
13  plot(T, Z, 'g');
14  [A B M] = bpoly(Ti, T, 20, 0.0001); % Sets up matrices needed by splinesmooth(). See bpoly()
15  newZ = splinesmooth(Delta, A, B, M, 0); % Smoothing with 0 regularization
16  plot(T, newZ+2, 'b--');
17  newZ = splinesmooth(Delta, A, B, M, 500000); % Smoothing with large regularization
18  plot(T, newZ+4, 'r--');
19  legend('original', 'lambda = 0', 'lambda = 500000');
20  title('Polynomial Basis');
21  hold off;
22  clear A
23  clear M

```

```

24 clear newZ
25
26 %%
27 figure; hold on;
28 plot(T, Z, 'g');
29 knots = [T(1)-2 T(1)-1 linspace(T(1), T(end), 10) T(end)+1 T(end)+2]; % knot vector
30 [A B M] = bspline(Ti, T, knots, 3, 0.0001); % Sets up matrices needed by splinesmooth()
31 newZ = splinesmooth(Delta, A, B, M, 0); % No regularization
32 plot(T, newZ+2, 'b--');
33 newZ = splinesmooth(Delta, A, B, M, 900000);
34 plot(T, newZ+4, 'r--');
35 legend('original', 'lambda = 0', 'lambda = 900000'); % Large regularization
36 title('B-Spline Basis');
37 hold off;

```

---

## Listing 6.2. Differences Smoothing Spline

---

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 % splinesmooth() fn for least squares differences fitting a signal. %
3 % Matrix set up for A, B, M are performed by bpoly(), bspline().%
4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5 function [ newZ ] = splinesmooth( Delta, A, B, M, Lambda )
6     % Delta: Observed differences
7     % A: Matrix of basis differences values (rows) over intersections Ti
8     % B: Matrix of basis values over all time
9     % M: Second derivative penalty matrix, the ijth entry
10    % corresponds to the integral of the second derivative of
11    % the ith basis times the second derivative of the jth

```



```

12      % basis over T.
13      % Lambda: Parameter enforcing smoothness condition. If set to 0,
14      % fit obtained is the ordinary least squares of the basis
15      % onto the data. As Lambda approaches infinity the
16      % solution converges to the least squares line fit.
17      Alpha = linsolve(A*A' + Lambda * M, A*Delta');
18      newZ = Alpha' * B; % Smoothing spline solution over a mesh. Same time dimension as B.
19  end

```

---

### Listing 6.3. Differences Polynomial Basis Evaluator

---

```

1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  % bpoly() sets up the matrices required by splinesmooth() fn %
3  % when performing a polynomial fit with a mesh of step size h %
4  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5  function [ A, B, M ] = bpoly( Ti, T, Degree, h )
6      % T: Time span for computing polynomial matrices
7      % Ti: Two dimensional matrix of intersection times
8      % Degree: The degree of the polynomial fit desired
9      % h: Step size for computing penalty matrix approximation
10     Tlower = Ti(1,:);
11     Tupper = Ti(2,:);
12     DiffT = T(1):h:T(end);
13     B = cell2mat(arrayfun(@(k) (T.^k)', 1:Degree, 'UniformOutput', false));
14     A = cell2mat(arrayfun(@(k) (Tupper.^k - Tlower.^k)', 1:Degree, 'UniformOutput', false));
15     K = cell2mat(arrayfun(@(k) (DiffT.^k)', 1:Degree, 'UniformOutput', false));
16     K = diff(K, 2, 2); % Approximates second derivative of basis elements over time span
17     M = K*K' ./ h^2; % Approximately computes the integrals, for penalty matrix M

```

18 **end**

---

Listing 6.4. Fast Differences Smoothing Spline Evaluator [6]

---

```
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 % bspline() uses bottom-up approach to set up the matrices required %
3 % by splinesmooth() when performing a spline fit with mesh step size h%
4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5 function [ A, B, M ] = bspline( Ti, T, Knots, Degree, h )
6     % Input:
7     % Ti Times on which intersections occurred, 2xl matrix
8     % T Time range for basis evaluation and penalization matrix
9     % Knots Vector of knots
10    % Degree Degree of basis elements
11    % h Mesh size for evaluation and taking second differences
12    % Output:
13    % A Matrix of basis differences values on intersections (rows)
14    % B Matrix of basis values over relevant times T
15    % M Second derivative penalty matrix, the ijth entry
16    % corresponds to the integral of the second derivative of
17    % the ith basis times the second derivative of the jth
18    % basis over T.
19    % See also bsplineslow
20    m = length(Knots);
21    Tlower = Ti(1,:); % Lower intersection times
22    Tupper = Ti(2,:); % Upper intersection times
23    DiffT = T(1):h:T(end); % Fine mesh for computation of second differences
24    basisT = zeros(m-1, length(T)); % Basis values on relevant times T
```

```

25 basisL = zeros(m-1, length(Ti)); % Basis values on Tlower of intersections
26 basisU = zeros(m-1, length(Ti)); % Basis values on Tupper of intersections
27 basisD = zeros(m-1, length(DiffT)); % Basis values on a fine mesh for second differences
28 H = @(t) t >= 0; % Heaviside helper function
29 % First we precompute all the 0 degree spline elements we will need in the time span
30 for j = 1:m-1
31     basisT(j,:) = H(T - Knots(j)) .* (1 - H(T - Knots(j+1)));
32     basisL(j,:) = H(Tlower - Knots(j)) .* (1 - H(Tlower - Knots(j+1)));
33     basisU(j,:) = H(Tupper - Knots(j)) .* (1 - H(Tupper - Knots(j+1)));
34     basisD(j,:) = H(DiffT - Knots(j)) .* (1 - H(DiffT - Knots(j+1)));
35 end
36 for n = 1:Degree % Now loop over the degrees until we reach the desired order
37     for j = 1:m-n-1 % Update new basis elements according to recursive formula
38         % Update basisT
39         f1 = basisT(j,:);
40         f2 = basisT(j+1,:);
41         basisT(j,:) = (T - Knots(j)) ./ (Knots(j+n) - Knots(j)) .* f1 + ...
42             (Knots(j+n+1) - T) ./ (Knots(j+n+1) - Knots(j+1)) .* f2;
43         % Update basisL
44         f1 = basisL(j,:);
45         f2 = basisL(j+1,:);
46         basisL(j,:) = (Tlower - Knots(j)) ./ (Knots(j+n) - Knots(j)) .* f1 + ...
47             (Knots(j+n+1) - Tlower) ./ (Knots(j+n+1) - Knots(j+1)) .* f2;
48         % Update basisU
49         f1 = basisU(j,:);
50         f2 = basisU(j+1,:);
51         basisU(j,:) = (Tupper - Knots(j)) ./ (Knots(j+n) - Knots(j)) .* f1 + ...

```

```

52         (Knots(j+n+1) - Tupper) ./ (Knots(j+n+1) - Knots(j+1)) .* f2;
53     % Update basisD
54     f1 = basisD(j,:);
55     f2 = basisD(j+1,:);
56     basisD(j,:) = (DiffT - Knots(j)) ./ (Knots(j+n) - Knots(j)) .* f1 + ...
57         (Knots(j+n+1) - DiffT) ./ (Knots(j+n+1) - Knots(j+1)) .* f2;
58     end
59 end
60 K = diff(basisD(1:m-1-Degree,:), 2, 2); % Approximate derivatives
61 A = basisU(1:m-1-Degree,:) - basisL(1:m-1-Degree,:);
62 B = basisT(1:m-1-Degree,:);
63 M = K * K' ./ h^2; % Approximate integrals for penalty matrix M
64 end

```

---

#### Listing 6.5. Slow Differences Smoothing Splines Evaluator

---

```

1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  % bsplineslow() returns a function handle for a spline. It uses a %
3  % naive evaluation strategy which leads to exponentially many %
4  % function calls. — Example usage: func_gen = bsplineslow(T) %
5  % my_spline = func_gen(2, 4) returns a function handle to the %
6  % second spline (with respect to the knot vector) of degree 4. %
7  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
8  function [ spline ] = bsplineslow( T )
9      % Input:
10     % T: Knot vector
11     %
12     % Output:

```

```

13      % spline-gen: A function which generates splines of arbitrary degree.
14      cache = {}; % Shares a little memory between recursive calls
15      H = @(t) t >= 0; % Heaviside function
16      for j = 1:length(T) - 1 % Populates the base cases for the spline
17          cache{j,1} = @(t) H(t - T(j)) .* (1 - H(t - T(j+1))); %#ok<AGROW>
18      end
19      function [ func ] = recursion( j, n ) % The spline generating function
20          if j + n > length(T)
21              error ('j + n must not exceed the number of control points.')
22          end
23          [l, k] = size(cache);
24          if j > l || n > k || isempty(cache{j,n}) % Checks to see if we have already cached this spline
25              f1 = recursion(j, n-1); % recursively retrieves lower degree spline definitions
26              f2 = recursion(j+1, n-1); % recursively retrieves lower degree spline definitions
27              % Applies recursive definition to compute unknown basis values
28              cache{j,n} = @(t) (t - T(j)) ./ (T(j+n-1) - T(j)) .* f1(t) + ...
29                  (T(j+n) - t) ./ (T(j+n) - T(j+1)) .* f2(t);
30          end
31          func = cache{j,n};
32      end
33      spline = @recursion;
34  end

```

---

Listing 6.6. Penalized Dictionary Inpainting Example Script

---

```

1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  % Penalized Dictionary Inpainting example script for how to use pdi()%
3  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

4  colormap gray
5  % Creating an artificial data set
6  I = [1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0;
7      0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1;
8      1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0;
9      0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1;
10     1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0;
11     0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1;
12     1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0;
13     0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1;
14     1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0;
15     0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1;
16     1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0;
17     0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1];
18 figure(1); imagesc(I); title('original');
19 lc = I;
20 lc(rand(size(I)) > .4) = nan; % Damaging a portion of it
21 lcdisplay = lc;
22 lcdisplay(isnan(lc)) = .5;
23 figure(2); imagesc(lcdisplay); title('damaged'); colormap gray
24 lp = pdi(lc, 6, 4, 0); % Reconstructing damaged image with pdi()
25 figure(3); imagesc(lp); title('reconstructed lambda = 0'); colormap gray
26 figure(4); imagesc(pdi(lc, 6, 4, 6)); title('reconstructed lambda = 6'); colormap gray

```

---

Listing 6.7. Penalized Dictionary Inpainting [18, 23]

---

```

1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %pdi() implements the Penalized Dictionary Inpainting Algorithm%

```

```

3  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4  function [lp, error] = pdi(l, p, k, lambda)
5      % Input:
6      % l: (Damaged) Image we wish to reconstruct
7      % p: Dimension of neighborhood (size of window). It is important that this value
8      % be comparable to the size of the structure in the images if we wish to
9      % effectively capture nonlocal information and complete patterns
10     % k: Number of singular vectors to keep. This controls for the complexity of the
11     % learned dictionary of pieces
12     %
13     % Output:
14     % lp: Inpainted image
15     % error: Error progression over iterations
16     fprintf('Populating neighborhood matrix ... ');
17     A = neighborhoods(l, p); % Creates neighborhood matrix from image l
18     fprintf('DONE\n');
19     tau = 0.003;
20     tol = 10^-2;
21     maxiter = 1000;
22     Mask = isfinite(A);
23     A(~Mask) = 0;
24     P = rand(size(A,1), k);
25     V = rand(k, size(A,2));
26     fprintf('Creating penalty matrix ... ');
27     M = penaltyMatrix(l, p); % Creates the penalty matrix for gradient descent
28     fprintf('DONE\n');
29     fprintf('Iterating ... \n')

```

```

30     counter = 1;
31     Delta = A - P*V .* Mask;
32     error = norm(Delta, 'fro');
33     % Perform gradient descent
34     while error(end) > tol && counter < maxiter
35         if mod(counter, 150) == 0 % Display progress occasionally
36             fprintf('Iteration %d: Error = %f\n', counter, error(end));
37         end
38         counter = counter + 1;
39         P = P + tau * Delta * V' - tau * lambda * M * P; % Update P
40         V = V + tau * P' * Delta; % Update V
41         Delta = (A - P*V) .* Mask; % Update Delta
42         error(end+1) = norm(Delta, 'fro');
43     end
44     B = P*V;
45     lp = reconstitute(B, l, p); % Recover image from neighborhood matrix representation
46     end
47
48     function [ A ] = neighborhoods(l, p)
49         % Runs a sliding window across image and creates serialization for neighborhood matrix
50         [h w] = size(l);
51         A = zeros((w-p+1) * (h-p+1), p^2);
52         for i = 1:h-p+1
53             for j = 1:p
54                 A((i-1)*(w-p+1)+1 : i*(w-p+1), j*p-p+1: j*p) = l(i:i+p-1, j:w-p+j)';
55             end
56         end

```



```

57 end
58
59 function [ M ] = penaltyMatrix(l, p)
60     % Creates penalty matrix corresponding to one-pixel shifted differences
61     % Edge cases are tricky.
62     [h w] = size(l);
63     maindiag = repmat([3 repmat(4, 1, w-p-1) 3], 1, h-p-1);
64     maindiag = [2 repmat(3, 1, w-p-1) 2 maindiag 2 repmat(3, 1, w-p-1) 2];
65     adjdiag = repmat([repmat(-1, 1, w-p) 0], 1, h-p+1);
66     farddiag = -ones(1, size(maindiag, 2));
67     M = spdiags([maindiag' adjdiag' adjdiag' farddiag' farddiag'], ...
68         [0 1 -1 (w-p+1) -(w-p+1)], (h-p+1)*(w-p+1), (h-p+1)*(w-p+1));
69 end
70
71 function [ lp ] = reconstitute(B, l, p)
72     % To recover image from neighborhood matrix B, we take the average pixel
73     % intensity across its various representations.
74     [h w] = size(l);
75     lp = zeros(h, w); % Initiate pixel intensity to 0
76     hits = zeros(size(lp)); % Number of times a pixel is represented, used for averaging
77     for i = 1:h-p+1
78         for j = 1:p
79             lp(i:i+p-1, j:w-p+j) = lp(i:i+p-1, j:w-p+j) + ...
80                 B((i-1)*(w-p+1)+1 : i*(w-p+1), j*p-p+1: j*p)'; % Increment pixel intensity
81             hits(i:i+p-1, j:w-p+j) = hits(i:i+p-1, j:w-p+j) + 1; % Increment representation count
82         end
83     end

```

```
84         lp = lp ./ hits; % take average (pixel-wise)
85     end
```

---

## BIBLIOGRAPHY

## BIBLIOGRAPHY

- [1] R. L. Burden and J. D. Faires. *Numerical Analysis (Eighth Edition)*, Thomson, 2005.
- [2] S.B. Andersson and D.Y. Abramovitch. “A Survey of Non-Raster Scan Methods With Application to Atomic Force Microscopy.” *Proceedings of American Control Conference*, pp. 3516–3521, 2007.
- [3] G. Aubert and P. Kornprobst. “Mathematical problems in image processing: Partial differential equations and the calculus of variations.” *Applied Mathematical Sciences*, vol. 147, 2006.
- [4] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. “Image inpainting.” *Proceedings of the 27th annual conference on Computer Graphics and Interactive Techniques*, SIGGRAPH ’00, pp. 417–424, 2000.
- [5] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher. “Simultaneous structure and texture image inpainting.” *Proceedings of Computer Vision and Pattern Recognition Conference*, vol. 2, pp. 707–712. IEEE Computer Society, June 2003.
- [6] C. De Boor. *A Practical Guide to Splines (Revised Edition)*, Springer, 2001.
- [7] K E. Atkinson. *An Introduction to Numerical Analysis (Second Edition)*, Wiley, 1988.
- [8] V. Caselles, J.M. Morel, and C. Sbert. “An axiomatic approach to image interpolation.” *Transactions on Image Processing*, pp. 376–386, March 1998.
- [9] T. Chan and J. Shen. “On the role of the bv image model in image restoration.” 2002.
- [10] P.I. Chang and S.B. Andersson. “Non-raster scanning in atomic force microscopy for high-speed imaging of biopolymers.” *Control Technologies for Emerging Micro and Nanoscale Systems*, Lecture Notes in Control and Information Sciences. Springer, 2011.
- [11] I. Chasiotis. “Atomic force microscopy in solid mechanics.” *Springer Handbook of Experimental Solid Mechanics*, Springer, 2008.
- [12] A. Chen, A. L. Bertozzi, P. D. Ashby, P. Getreuer, and Y. Lou. “Enhancement and recovery in atomic force microscopy images.” 2011.

- [13] A. I.-A. Chen, J. Huang, and S. Lim. “Multi-point boundary tracking for atomic force microscopy.” *Technical report, University of California Los Angeles, Department of Mathematics*, 2009.
- [14] D. Croft, G. Shed, and S. Devasia. “Creep, hysteresis, and vibration compensation for piezoactuators: Atomic force microscopy applications.” *Journal of Dynamic Systems, Measurement, and Control*, vol. 1, pp. 35–43, 2001.
- [15] M. D’Acunto and O. Salvetti. “Pattern recognition methods for thermal drift correction in atomic force microscopy imaging.” *Pattern Recognition and Image Analysis*, vol. 21, pp. 9–19, March 2011.
- [16] S. Devasia, E. Eleftheriou, and S. Moheimani. “A survey of control issues in nanopositioning.” *IEEE Transactions on Control Systems Technology*, vol. 5, pp. 802–823, September 2007.
- [17] R. A. Finkel and J.L. Bentley. “Quad trees a data structure for retrieval on composite keys.” *Acta Informatica*, vol. 4, pp. 1–9, 1974.
- [18] Simon Funk. *Netflix update: Try this at home*, December 2006.
- [19] T. Goldstein and S. Osher. “The split Bregman method for L1-regularized problems.” *SIAM Journal on Imaging Sciences*, vol. 2, pp. 323–343, 2009.
- [20] J. Huang and K. Thompson. “Atomic force microscopy: Algorithms for efficient imaging.” *Technical report, University of California Los Angeles, Department of Mathematics*, August 2010.
- [21] I. A. Mahmood and S. O. R. Moheimani. “Fast spiral-scan atomic force microscopy.” *Nanotechnology*, 20(36):365503, August 2009.
- [22] I. A. Mahmood, S. O. R. Moheimani, and B. Bhikkaji. “A new scanning method for fast atomic force microscopy.” *Transactions on Nanotechnology*, vol. 2, pp. 203–216, March 2011.
- [23] T. Meyer. “Non-local rank-reduction image processing.” *Technical report, University of California Los Angeles, Department of Mathematics*, January 2011.
- [24] R. A. Horn and C. R. Johnson. *Matrix Analysis*, Cambridge University Press, 2009.
- [25] B. Mokaberi and A. Requicha. “Drift compensation for automatic nanomanipulation with scanning probe microscopes.” *Transactions on Automation Science and Engineering*, vol. 3, pp. 199–207, July 2006.

- [26] H. Prautzsch, W. Boehm, and M. Paluszny. *Bézier and b-spline techniques*. Springer-Verlag, October 2002.
- [27] L. M. Howser R. E. Smith, J. M. Price. “A smoothing algorithm using cubic spline functions.” *Technical report, National Aeronautics and Space Administration*, 1974.
- [28] C. H. Reinsch. “Smoothing by spline functions.” *Numerische Mathematik*, February 1967.
- [29] M. Rost, G. van Baarle, A. Katan, W. van Spengen, P. Schakel, W. van Loo, T. Oosterkamp, and J. Frenken. “Video-rate scanning probe control challenges: setting the stage for a microscopy revolution.” *Asian Journal of Control*, vol. 2, pp. 110–129, 2009.
- [30] R. Meka, P. Jain, I. S. Dhillon. “Guaranteed Rank Minimization via Singular Value Projection.” *Neural Information Processing Systems*, 2009.
- [31] B. Salmons, D. Katz, and M. Trawick. “Correction of distortion due to thermal drift in scanning probe microscopy.” *Ultramicroscopy*, pp. 339–349, 2010.
- [32] M. Kirby. *Geometric Data Analysis: An Empirical Approach to Dimensionality Reduction and the Study of Patterns*, Wiley, 2001.
- [33] J. Tumblin and G. Turk. “LCIS: A boundary hierarchy for detail-preserving contrast reduction.” *Proceedings of the 26th annual conference on Computer Graphics and Interactive Techniques*, pp. 83–90. ACM Press/Addison-Wesley Publishing Co., 1999.
- [34] Y. K. Yong, S. O. R. Moheimani, and I. R. Petersen. “High-speed cycloid-scan atomic force microscopy.” *Nanotechnology*, 21(36):365503, August 2010.
- [35] ”Atomic force microscopy.” Wikipedia, The Free Encyclopedia. Wikimedia Foundation, Inc. 22 July 2004.