Do the Time-Warp: Continuous Alignment of Gene Expression Time-Series Data

by

Georg Kurt Gerber

B.A., Mathematics, UC Berkeley (1991)
M.P.H., Infectious Diseases, UC Berkeley (1994)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2003

Author .....
Department of Electrical Engineering and Computer Science
September 1, 2002

Certified by .....
David K. Gifford
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Do the Time-Warp: Continuous Alignment of Gene Expression Time-Series Data

by

Georg Kurt Gerber

Submitted to the Department of Electrical Engineering and Computer Science
on September 1, 2002, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

Recent advances in DNA microarray technologies are enabling researchers to measure the expression levels of thousands of genes simultaneously, with time-series data offering particularly rich opportunities for understanding dynamic biological processes. Unfortunately, DNA microarray data is plagued by quality issues including noise and missing data points. Expression time-series data introduce additional complications, including sampling rate differences between experiments and variations in the timing of biological processes.

In this thesis, a probabilistic framework is presented for making gene expression time-series data from different experiments directly comparable. This method addresses the issues of DNA microarray data noise, missing values, different sampling rates between experiments, and variability in the timing of biological processes. Gene expression time-series data are represented as a set of continuous spline curves (piecewise polynomials) using a linear mixed-effects model, a statistical method that allows for a fixed or group effect as well as a random or individual effect. The model constrains a gene's expression curve to be generally similar to those of co-expressed genes, while still allowing for gene specific variations. A continuous alignment algorithm is then introduced in which a search is performed for a function that transforms the time-scale of one gene expression process into that of another, while minimizing a measure of error between expression profiles in the two processes.

Three applications are then presented of the continuous representation/alignment algorithm to DNA microarray expression data, which yield biologically meaningful results. First, it is shown that the linear mixed-effects spline model can effectively predict missing data values, achieving lower error for cases of one or two consecutive missing values than the best method previously reported in the literature. Second, it is demonstrated that the alignment algorithm can produce stable low-error alignments, allowing data from three different yeast cell cycle experiments (with period and phase differences) to be combined. Finally, it is shown that the alignment algorithm can be used to identify genes that behave differently in normal versus genetically modified cells.

Thesis Supervisor: David K. Gifford
Title: Professor of Electrical Engineering and Computer Science

Biographical Information

Personal

Born: December 31, 1970 in Los Angeles, CA.

Education

B.A., Mathematics, *summa cum laude*, UC Berkeley, 1991.
M.P.H., Infectious Diseases, UC Berkeley, 1994.

Work Experience

- Graduate Research Assistant, M.I.T. Laboratory for Computer Science (Supervisor: Prof. David Gifford), 2000-present.
- Instructor, Northeastern University Bioinformatics Essentials Graduate Certificate Program, 2002.
- Chief Technology Officer, IS?TV, Inc., 2000.
- Senior Vice President, L-Squared Entertainment, Inc., 1995-1999.
- President and Chief Executive Officer, Infinite Interactions, Inc., 1993-1995.
- Graduate Research Assistant, UC Berkeley Department of Computer Science (Supervisor: Prof. Brian Barsky), 1993-1994.
- Research Associate/Laboratory Automations Specialist, Metabolex, Inc., 1992.
- Undergraduate Research Assistant, UC Berkeley Department of Biophysics (Supervisor: Prof. Hans Bremermann), 1990.
- President, Speak Easy Communications System, 1984-1989.

Awards

- National Defense Science and Engineering Graduate (NDSEG) Fellowship (awarded and accepted), 2002.
- National Science Foundation (NSF) Fellowship (awarded), 2002.
- University of California Regents' Graduate Fellowship, 1992-1994.
- UC Berkeley Highest Distinction in General Scholarship in Mathematics, 1991.
- Phi Beta Kappa honor society, elected 1990.
- University of California Regents' Undergraduate Scholarship, 1989-1991.
- UC Berkeley Dean's List, every semester attended, 1989-1991.
- J&M Seitz Scholarship, 1989.

Acknowledgments

I must first thank Ziv-Bar Joseph, a fellow graduate student in my research group and collaborator on much of the work that led to this thesis. Ziv was instrumental in adapting the linear mixed-effects spline model for use on gene expression time-series data. He was also very helpful in many other aspects of this work, such as finding applications and coming up with validation techniques. Ziv has been more than just a great intellectual collaborator; he has been a true friend.

I must also acknowledge Professor David Gifford, my research supervisor. He first introduced me to the field of computational functional genomics and has been instrumental in encouraging my research by providing ideas, resources, and guidance while also fostering the critical collaborative relationships outside of our immediate group. Professor Gifford is one of those rare individual who possesses both the intellectual power and dedication needed to crack hard research problems, as well as the courage and entrepreneurial spirit needed to blaze trails in a new field.

Professor Tommi Jaakkola has also been critical in the development of this thesis. He first suggested applying mixed-effects spline models to gene expression data, and subsequently provided insights and guidance throughout this entire research project. Professor Jaakkola's deep knowledge of statistics and machine learning has inspired my own interests in these subjects.

Professor Rick Young at the Whitehead Institute has provided much inspiration for the "biological half" of this thesis. His passion for biology and unlocking the secrets of genetic regulation is infectious. Professor Young has also been critical in fostering true collaborations between biologists in his lab and computer scientists in my group. I must specifically thank him for encouraging my relationship with Dr. Itamar Simon, a Post-Doctoral Researcher in the Young lab. Dr. Simon's work on the cell cycle genetic regulatory network provided much of the initial motivation for the algorithms presented in this thesis. He also suggested the application of the alignment algorithm to identifying genes that are differentially expressed in wildtype versus mutant cells.

Within my own research group, I must acknowledge fellow graduate students Reina Riemann, John Barnett, and Tim Danford for the many stimulating ideas about computational biology that they have shared with me. Reina Riemann, my office mate during my first year with the group, was particularly helpful in providing insights during the early stages of the work that led to this thesis. Jeanne Darling, Professor Gifford's secretary, is one of the most knowledgeable and helpful people I have encountered at M.I.T. (or anywhere else for that matter!) She has always been there to offer advice and to lend a helping hand as I have tried to find my way through the M.I.T. maze.

I would also like to acknowledge the memories of two teachers from my pre-M.I.T. days who were critical in my intellectual development. Professor Donald Mazukelli introduced me to the beauty of mathematics, giving me a life-long gift. Professor Hans Bremermann, one of the humblest and warmest people I have ever known, introduced me to the concept of trying to use equations to describe the behavior of living organisms, and in so doing, sparked my unrelenting passion for mathematical biology.

Finally, I must acknowledge my family, who started setting me on my intellectual path before I even knew it. My brother Karl grew up with me like a twin, and his determination and risk-taking spirit rubbed off on me. My older sister Margot has always supported my offbeat interests and encouraged me to think outside the box. Although my father Barry has never had much tolerance for moldering academics, since my earliest memories he has

always conveyed to me a great love of learning. He never separated me from "adult" ideas or tools, always letting me explore his books, computers, and electronic gadgets. Most importantly, he had the wisdom to do something incredibly hard: he resisted intruding too much in my explorations so that I could learn to think independently. My mother Jane sparked my curiosity and love for living things. When I was young, she spent endless hours reading books about dinosaurs and pre-historic people to me, radiating her sense of the mystery and beauty of the biological world. Years later, she revealed another trait — courage — when she fought a serious illness, and once again reminded me of just how important life is.

Finally, I must acknowledge Anne-Marie (1995-2002), for being a part of my life for as long as she could. She was there throughout so much, both the happy and the sad: our days in Los Angeles when I worked as an executive in Hollywood, my mothers' illness, my difficult decision to re-enter the academic world, and our move across the country to Boston. I miss her very much, and will always remember the love and happy times we enjoyed together.

A Note on Notation

An effort has been made to use consistent notation throughout this thesis. Scalars will be denoted with italic Roman or Greek lower-case letters such as x and y or μ. All vectors will be column vectors and denoted with bold italic Roman or Greek lower-case letters such as $\boldsymbol{x}$ and $\boldsymbol{y}$ or $\boldsymbol{\mu}$. Matrices will be denoted with bold italic Roman or Greek upper-case letters such as $\boldsymbol{A}$ and $\boldsymbol{B}$ or $\boldsymbol{\Gamma}$. Sets will be denoted with upper-case Roman italic letters such as C and T. Random variables will be denoted by fonts without serifs such as x and y for random scalars and $\mathbf{x}$ and $\mathbf{y}$ for random vectors. Unfortunately, lower-case Greek letters are also used for random variables but the sans serif style is not available. In these cases, the intention should be clear from the context.

"...he ended up recommending to all of them that they leave Macondo, that they forget everything he had taught them about the world and the human heart, that they shit on Horace, and that wherever they might be they always remember that the past was a lie, that memory has no return, that every spring gone by could never be recovered, and that the wildest and most tenacious love was an ephemeral truth in the end."

— Gabriel Garcia Marquez, *One Hundred Years of Solitude*

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Description

Recent advances in DNA microarray technologies are enabling researchers to measure the expression levels of thousands of genes simultaneously. These technologies are proving invaluable in functional genomics, a developing field in which investigators ultimately seek to understand genetic activity at the level of an organism's entire collection of genes [25]. Time-series expression data, the focus of this thesis, offer particularly rich opportunities for understanding dynamic biological processes such as the cell cycle, development, and immune response [51, 57, 38].

Two common analytical objectives in functional genomics are, 1) identifying genes that behave differently under alternate experimental conditions and, 2) building networks of genetic interactions. In the first case, researchers may be interested in determining which genes are affected by an experimental intervention such as a gene knockout [60], or they may want to identify genes whose expression patterns are altered during a natural pathological process such as cancer [4]. The goal in such experiments is to identify significant differences between gene expression in a "normal" versus an alternate condition. In the second type of analysis, researchers seek to construct networks of genetic interactions and to understand complexities of these system, such as the role of combinatorial regulation or feedback loops. Given the scale and the complexity of the biological systems being studied, sophisticated computational techniques such as Bayesian Networks have been employed [19, 21, 22].

Unfortunately, DNA microarray data are plagued with quality issues that make both of the above mentioned analyses challenging. Large amounts of noise may be introduced during the many steps of sample preparation, hybridization, and detection. Further, in some experiments expression data points for a given gene may be "missing", due to practical difficulties such as detergent bubbles on glass slides or manufacturing defects in the microarray [23]. For these reasons, a large literature has developed describing computational and experimental methods for dealing with the noise inherent in DNA microarray data (see for instance [26, 54, 12, 29, 13]). However, few of these methods address the specific features of temporal microarray data.

On the one hand, gene expression time-series data provide information lacking in static data that can be useful in overcoming noise and missing value problems. For instance, it should be possible to exploit the reasonable assumptions that adjacent time points will be statistically dependent on one another, and that expression values will vary fairly smoothly with time. On the other hand, gene expression time-series data introduce two complications

not present in static analyses: lack of standard sampling rates and variability in the *timing* of biological processes.

The first issue is a practical one: since DNA microarrays are expensive and time-series experiments are often technically challenging to perform, biologists want to sample a minimum number of time-points. Unfortunately, there is a dearth of empirical or theoretical information on appropriate sampling rates, since large-scale gene expression time-series experiments are a relatively new phenomena. This has resulted in published DNA microarray studies of the same or similar biological processes sampled at very different time-points. Non-uniform sampling or coverage of different portions of the process add to the difficulties in comparing data sets.

The second issue is a biological one: the rate at which similar underlying processes unfold can be expected to differ across organisms, genetic variants and environmental conditions. For instance, extensive gene expression time-series data has been collected to study the yeast cell cycle [51]. In these experiments, different methods have been used to synchronize the cells, and these have resulted in variations in growth rates and other aspects of the cell cycle. Thus, while the data sets collected capture the same fundamental process, the phases and periods of the expression levels of cell cycle genes differ markedly between the experiments.

When analyzing time-series data for the objectives of identifying differentially expressed genes and building genetic networks, one must address both the issue of different sampling rates between experiments and that of biological variability in the timing of processes. In the case of identifying differentially expressed genes, one would like to analyze data from the two conditions on the same time-scale so that direct comparisons can be made. For the second objective of building genetic networks, the statistical models often used require large amounts of data. One would like to combine as much data measuring the same or similar biological processes as possible. Thus, for both analytical objectives one would like a computational method for making gene expression time-series data from different experiments directly comparable. Further, the method should exploit the special structure of time-series data to deal with the general problems of DNA microarray data — noise and missing data points.

1.2 Solution Overview

In this thesis, a probabilistic framework will be presented for making gene expression time-series data from different experiments directly comparable. This method addresses the issues of DNA microarray data noise, missing values, different sampling rates between experiments, and variability in the timing of biological processes. Gene expression time-series data are represented as a set of continuous curves whose time-scales can be smoothly warped so as to align different data sets. Splines or piecewise polynomials are a natural tool for estimating such smooth curves. However, given the noise and missing value problems inherent in DNA microarray data, fitting an independent spline to each gene's expression pattern can be problematic. Fortunately, biology can help: it has been demonstrated that genes involved in similar processes often do not behave independently [15]. Such genes are said to be *co-expressed*, and can be identified using previous biological knowledge or clustering algorithms. The expression profiles of co-expressed genes can be assumed to be statistically dependent on one another, but aren't expected to be identical. For instance, co-expressed genes might be expected to peak at approximately the same time, but the magnitudes of the peaks could

differ somewhat.

The statistical model presented in this thesis uses co-expression information to constrain the shape of a gene's expression profile (spline) to be generally similar to those of co-expressed genes, while still allowing for gene specific variations. Further, these similarity constraints are lessened when more information is available, i.e., there are a large number of data points for a gene's expression time-series. The type of model used is a linear mixed-effects model, which assumes that observations of a given random variable can be explained as a combination of a *fixed* or group effect, as well as an individual or *random* effect [30, 27]. Due to the presence of the random effects variables and shared parameters, the parameters of the linear mixed-effects model cannot be estimated using a closed form solution. However, the structure of the model makes the problem amenable to approximate solution using the Expectation-Maximization (EM) algorithm.

A method for aligning sets of continuous gene expression profiles from two different experiments is then presented. The core idea behind this method is to search for a function that transforms the time-scale of one gene expression process into that of another, while minimizing a measure of error between expression profiles in the two processes. Because a parameterized function is used, the allowed number of degrees of freedom is explicitly specified, which is helpful in avoiding overfitting. The problem of finding the parameters that minimize the alignment error measure is in general a non-linear constrained optimization problem and numerical methods must be used. Since the alignment algorithm may need to handle a large number of genes, a stochastic optimization method that can reduce the computation time is presented. Further, a closed-form solution that can speed up the alignment algorithm in a special case is also described.

1.3 Thesis Roadmap

The remainder of this thesis will be organized as follows. In the next section of this chapter, related work will be discussed.

The probabilistic model for estimating gene expression time-series data with continuous curves will then be presented in Chapter 2, along with necessary background information including an overview of linear mixed-effects models, the Expectation-Maximization (EM) algorithm, and splines.

In Chapter 3 the framework for aligning continuous gene expression time-series profiles will be presented. The algorithm used to search for minimizing parameters in the alignment will also be discussed, as well as some methods for speeding up the alignment algorithm on large data sets.

In Chapter 4, three applications of the continuous representation/alignment algorithms to DNA microarray expression data will be presented. In the first application, it will be shown that the linear mixed-effects spline model introduced in Chapter 2 can effectively predict missing data values, achieving lower error for cases of one or two consecutive missing values than the best method previously reported in the literature. Second, it will be demonstrated that the alignment algorithm can produce stable low-error alignments, allowing data from three different yeast cell cycle experiments (with period and phase differences) to be combined. Finally, it will be shown that the alignment algorithm can be used to identify genes that behave differently in normal versus genetically modified cells.

In Chapter 5, a brief summary of this thesis will be presented as well as some ideas for future work. Those unfamiliar with biology and microarrays may want to read Appendices A

and B before proceeding with the rest of this thesis. In Appendix A, the basics of molecular genetics are reviewed and in Appendix B, DNA microarrays and some of the limitations of these technologies are discussed. Finally, in Appendix C derivations are provided of some of the formulas related to the linear mixed-effects spline model for gene expression time-series data.

1.4 Related Work

Much of this work was previously published in a conference paper that I co-authored [6]. This thesis focuses on my particular contributions to this earlier work, and provides considerably more detail on the background and computational methods used.

There is a considerable statistical literature in which mixed-effects spline models are applied to non-uniformly sampled time-series data. James and Hastie [27] presented a reduced rank model that was used for classifying medical time-series data. The spline model presented in this thesis is based on their work. However, since this thesis deals with the alignment of gene expression time-series data, the focus is on predicting individual spline curves for genes rather than determining an underlying class profile as in [27]. Another difference between this work and theirs is that a reduced rank approach is not necessary, since gene expression data sets contain information about thousands of "cases" (genes).

D'haeseleer [11] used spline interpolation on expression time-series data for individual genes to estimate missing values. As shown in section 4.2, this technique cannot approximate the expression levels of a gene well, especially if there are several consecutive missing values.

Several recent papers have dealt with modelling and analyzing gene expression time-series data. Zhao *et al* [59] analyze yeast cell cycle data, fitting a statistical model to gene expression profiles that is custom-tailored for periodic data. Ramoni *et al* [43] model gene expression time-series as autoregressive processes and apply a Bayesian framework for clustering genes. Both Zhao *et al* and Ramoni *et al* do not deal with alignment, and their methods are less general than the one presented here, since they make strong assumptions about underlying statistical models. Further, it is unclear how data sampled at different rates could be compared within their frameworks.

Reis *et al* [46] use the first-order difference (slope) vector of gene expression time-series to build correlation based relevance networks. Their method does not deal with alignment or present a statistical model, but rather defines a distance measure that they use for finding associations between gene pairs.

Aach *et al* [1] presented a method for aligning gene expression time-series data that is based on Dynamic Time Warping, a discrete method that uses dynamic programming and is conceptually similar to sequence alignment algorithms. Unlike with the method presented in this thesis, the allowed degrees of freedom of the warp operation in Aach *et al* depends on the number of data points in the time-series. Their algorithm also allows mapping of multiple time-points to a single point, thus "stopping time" in one of the data sets. In contrast, the algorithm presented in this thesis avoids temporal discontinuities by using a continuous warping representation.

Qian *et al* [42] use a discrete sequence alignment-type algorithm similar to that in Aach *et al*. Their goal is to define a distance measure that allows for a phase difference between genes' expression patterns and to use this for clustering. They align individual genes rather than sets, which may lead to low quality alignments. Further, their method allows only for phase shifts and all genes must be sampled at the same time-points.

There is a substantial body of work from the speech recognition and computer vision community that deals with data alignment. For instance, non-stationary Hidden Markov models with warping parameters have been used for alignment of speech data [10], and mutual information based methods have been used for registering medical images [55]. However, these methods generally assume data are of high resolution, which is not the case with available gene expression data sets.

Ramsay and Li [45] present a continuous alignment framework that is conceptually similar to the one in this thesis, but theirs is not applied to gene expression data and there is no underlying statistical model used to fit discrete data. Also, with their method, curves must begin and end at the same point, which does not allow for partial overlaps as does the method presented here. Further, they do not describe methods for differential weighting of alignment errors, or for speeding up the algorithm so that it operates efficiently on large data sets.

Chapter 2

A Probabilistic Model for Estimating Gene Expression Time-Series Data with Continuous Curves

The purpose of this chapter is to develop the necessary background, and then to present a statistical model for representing gene expression time-series data with continuous curves. The appropriateness of a continuous model rests on two key assumptions: adjacent gene expression time-points are statistically dependent on one another, and that expression values vary fairly smoothly with time. The first assumption of statistical dependence is unlikely to be controversial, since time-series experiments are essentially designed with this objective in mind. The second assumption of smooth variation of expression data across time is more difficult to verify empirically, due to quality issues inherent in DNA microarray data. But, the argument can be made on more theoretical grounds. The most important point is that DNA microarrays measure mRNA levels of *populations* of cells. Thus, while it is true that gene expression is a stochastic process involving individual molecules [5], current microarray technologies only allow us to see an effect averaged over a large number of cells. This averaging can be reasonably expected to smooth out any dramatic variations in expression of a single gene over a short time scale.

Splines or piecewise polynomials are frequently used for fitting smooth curves to discrete observations [16]. However, noise and missing value problems inherent in DNA microarray data make fitting an independent spline to each gene's expression pattern problematic. Fortunately, biology can help: it has been demonstrated that genes involved in similar processes often do not behave independently [15]. Such genes are said to be *co-expressed*, and can be identified using previous biological knowledge or clustering algorithms. The expression profiles of co-expressed genes can be assumed to be statistically dependent on one another, but aren't expected to be identical. For instance, co-expressed genes might be expected to peak at approximately the same time, but the magnitudes of the peaks could differ somewhat.

An appropriate statistical framework for capturing this type of behavior is a linear *mixed-effects* model. With such a model, it is possible to constrain the shape of a gene's expression profile to be generally similar to those of co-expressed genes, while still allowing for gene specific variations. Further, these similarity constraints are lessened when more

information is available, i.e., there are a large number of data points for a gene's expression time-series.

The remainder of this chapter is organized as follows. First, background information on linear mixed-effects models is presented. Next, parameter estimation for these models using the Expectation-Maximization (EM) Algorithm is discussed. Then, a brief overview of splines is given. Finally, the linear mixed-effects spline model for gene expression time-series data is presented.

2.1 Linear Mixed-Effects Models

A linear mixed-effects model is a statistical model that assumes that observations of a given random variable can be explained as a combination of a *fixed* or group effect as well as an individual or *random* effect [30, 28, 27]. As a concrete example, suppose a study is conducted in which the body temperatures of a group of boys are measured weekly for a year. Further suppose that these boys live together in a group home, and are of similar ages and come from the same ethnic background. Given this experimental set-up, one would expect that at any particular time, the boys would have similar temperatures. Their temperatures might even be expected to fluctuate together, e.g., to be higher during flu season. However, we would also expect that some boys would have systematically higher or lower temperatures, due to individual differences such as genetics or immunity acquired from prior exposure to infections. Gene expression time-series data also has these features: co-expressed genes are expected to share some commonalities, but individual genes may show systematic expression differences.

A linear mixed-effects model can be described formally as follows. Suppose we have a set of m-dimensional random vectors each denoted by $\mathbf{y}_i$, where $i = 1 \ldots n$, e.g., n individuals each observed at m time-points. Let $\boldsymbol{\gamma}_i$ denote a random effects variable for each individual i, where each $\boldsymbol{\gamma}_i$ is a q-dimensional normally distributed random vector with mean zero and covariance matrix $\boldsymbol{\Gamma}$. Let $\boldsymbol{\mu}$ denote a q-dimensional vector of the fixed effects parameters. Denote by $\boldsymbol{A}_i$ and $\boldsymbol{B}_i$ the m by q (pre-determined) parameter matrices for each individual i. Finally, let $\boldsymbol{\epsilon}_i$ be a "noise" term, an m-dimensional normally distributed random vector with mean zero and covariance matrix $\sigma^2 \boldsymbol{I}_{mXm}$. Note that the parameters $\boldsymbol{\mu}$, $\boldsymbol{\Gamma}$ and σ are shared by all individuals. We can then write the linear mixed-effects model as:

$$\mathbf{y}_i = \boldsymbol{A}_i \boldsymbol{\mu} + \boldsymbol{B}_i \boldsymbol{\gamma}_i + \boldsymbol{\epsilon}_i \tag{2.1}$$

For convenience, let $\boldsymbol{\Psi} = (\boldsymbol{\mu}, \boldsymbol{\Gamma}, \sigma)^T$ denote a vector of all the parameters that we will estimate. It will also be assumed that variables $\mathbf{y}_i$ are independent given the parameters $\boldsymbol{\Psi}$, i.e., $p(\boldsymbol{y}_i, \boldsymbol{y}_j; \boldsymbol{\Psi}) = p(\boldsymbol{y}_i; \boldsymbol{\Psi}) p(\boldsymbol{y}_j; \boldsymbol{\Psi})$ for $i \neq j$. The same assumption will also be made about the random effects variables, i.e., $p(\boldsymbol{\gamma}_i, \boldsymbol{\gamma}_j; \Psi) = p(\boldsymbol{\gamma}_i; \boldsymbol{\Psi}) p(\boldsymbol{\gamma}_j; \boldsymbol{\Psi})$ for $i \neq j$. This essentially means that we are assuming that measurements of the variables pertaining to the n "individuals" were obtained independently. While this is clearly not entirely realistic, more complex dependencies are difficult to model and disambiguate, and may represent mostly secondary effects.

As can be seen from equation 2.1, each $\mathbf{y}_i$ is explained by a combination of a shared fixed effect $\boldsymbol{\mu}$, an individual-specific random effect $\boldsymbol{\gamma}_i$, and an additive noise term $\boldsymbol{\epsilon}_i$. Each $\mathbf{y}_i$ can be seen to be normally distributed with mean $\boldsymbol{A}_i \boldsymbol{\mu}$ and variance that depends on

both the random effect and noise term:

$$\text{var}(\mathbf{y}_i) = E[(\mathbf{y}_i - \boldsymbol{A}_i\boldsymbol{\mu})(\mathbf{y}_i - \boldsymbol{A}_i\boldsymbol{\mu})^T] =$$
$$E[(\boldsymbol{B}_i\boldsymbol{\gamma}_i + \boldsymbol{\epsilon}_i)(\boldsymbol{B}_i\boldsymbol{\gamma}_i + \boldsymbol{\epsilon}_i)^T] = \boldsymbol{B}_i\boldsymbol{\Gamma}\boldsymbol{B}_i^T + \sigma^2\boldsymbol{I}$$

Note that the conditional distribution of $\mathbf{y}_i$ given $\boldsymbol{\gamma}_i$ is particularly simple:

$$\mathbf{y}_i|\boldsymbol{\gamma}_i \sim N(\boldsymbol{A}_i\boldsymbol{\mu} + \boldsymbol{B}_i\boldsymbol{\gamma}_i, \sigma^2\boldsymbol{I})$$

That is, $\mathbf{y}_i|\boldsymbol{\gamma}_i$ is a normally distributed variable with a mean combining the fixed and random effect and variance dependent only on the noise term.

A common objective with a linear mixed-effects model is to *predict* the random effect for each variable conditioned on available observed data [30]. For instance, in the previous example of measuring boys' temperatures, one might like to estimate how much systematically higher or lower a given boy's temperature is than that of others in the group. If we use just the temperature measurements for that particular boy, we are throwing potential knowledge away — we are discarding information common to all the boys such as seasonal temperature variations due to illnesses, or noise in the measurements due to thermometer inaccuracy. Intuitively, we would somehow like to use *all* the data to make our prediction.

To be more precise, suppose we have observed an m-dimensional data vector $\boldsymbol{d}_i$ for each random variable $\mathbf{y}_i$. It can be proved that the *best predictor* of a random effects variable[1], denoted $\widehat{\boldsymbol{\gamma}}_i$, can be written as (see [58] for details):

$$\widehat{\boldsymbol{\gamma}}_i = E_{\boldsymbol{\Psi}}[\boldsymbol{\gamma}_i|\mathbf{y}_i] = \int \boldsymbol{\gamma}_i p(\boldsymbol{\gamma}_i|\boldsymbol{y}_i = \boldsymbol{d_i}; \boldsymbol{\Psi}) d\boldsymbol{\gamma}_i$$

Note that since the random variables are linked through a common group mean, covariance, and noise parameters, each random effects predictor $\widehat{\boldsymbol{\gamma}}_i$ will make use of all the available data. A closed form expression can be derived for the random effects predictor (see section C.1 for details):

$$\widehat{\boldsymbol{\gamma}}_i = \boldsymbol{\Gamma}\boldsymbol{B}_i^T(\boldsymbol{B}_i\boldsymbol{\Gamma}\boldsymbol{B}_i^T + \sigma^2\boldsymbol{I})^{-1}(\boldsymbol{d}_i - \boldsymbol{A}_i\boldsymbol{\mu}) \tag{2.2}$$

Equation 2.2 has a nice intuitive interpretation. Since the prior distribution of $\boldsymbol{\gamma}_i$ is $N(\mathbf{0}, \boldsymbol{\Gamma})$, our prediction for the random effect in the absence of data is simply $E[\boldsymbol{\gamma}_i] = \mathbf{0}$. The term $(\boldsymbol{d}_i - \boldsymbol{A}_i\boldsymbol{\mu})$ in equation 2.2 can be interpreted as the difference between the observation vector $\boldsymbol{d}_i$ and the best prior prediction of this vector ($E[\mathbf{y}_i] = \boldsymbol{A}_i\boldsymbol{\mu}$). The matrix $\boldsymbol{\Gamma}\boldsymbol{B}_i^T(\boldsymbol{B}_i\boldsymbol{\Gamma}\boldsymbol{B}_i^T + \sigma^2\boldsymbol{I})^{-1}$ can be interpreted as a *gain* matrix that controls the weight of the new information obtained from the observation versus the prior information. As the covariance matrix $\boldsymbol{\Gamma}$ increases (in the sense of its trace), we are less certain about the prior information about $\boldsymbol{\gamma}_i$ and the gain matrix weights up the observed data. The reverse occurs as the covariance matrix decreases and the prior information becomes stronger.

In order to use equation 2.2, we will need to estimate the parameters $\boldsymbol{\mu}$, $\boldsymbol{\Gamma}$, and σ. A standard approach to this type of problem is to maximize the likelihood of the data with respect to the unknown parameters. That is, let $\mathbf{y}$ denote a random vector concatenating all the $\mathbf{y}_i$'s. Thus, if $p(\boldsymbol{y}; \boldsymbol{\Psi})$ is the probability density function (p.d.f) for $\mathbf{y}$ we can form the data likelihood $L(\boldsymbol{\Psi}) = p(\boldsymbol{y}; \boldsymbol{\Psi})$ and our objective is to find:

$$\widehat{\boldsymbol{\Psi}} = \arg\max_{\boldsymbol{\Psi}}\{L(\boldsymbol{\Psi})\} \tag{2.3}$$

[1]This is sometimes called the best linear unbiased predictor (BLUP).

Due to the presence of the random effects variables and the shared parameters, this problem cannot be solved in closed form. Fortunately, the structure of the linear mixed-effects model makes it amenable to approximate solution using the Expectation-Maximization (EM) algorithm.

2.2 The Expectation-Maximization (EM) Algorithm

The Expectation-Maximization (EM) algorithm is a general method for iteratively computing maximum likelihood (ML) estimates. There is an extensive applied and theoretical literature on this topic. See the text by McLachlan and Krishnan for a good overview [35]. The discussion here will be restricted to the essentials needed to solve the ML problem for the linear mixed-effects model as discussion in section 2.1.

The EM algorithm exploits the idea that certain types of maximum likelihood estimation problems could be solved readily if additional data were available. To be more precise, let $\mathbf{y}$ denote a random vector from which a vector of observed data $\boldsymbol{d}$ has been sampled. Let $p(\boldsymbol{y};\boldsymbol{\Psi})$ denote the p.d.f for $\mathbf{y}$, and $L(\boldsymbol{\Psi}) = p(\boldsymbol{y};\boldsymbol{\Psi})$ denote the data likelihood function, where $\boldsymbol{\Psi}$ is a vector of parameters. Now, let $\mathbf{z}$ denote a random vector of unobserved or "missing" data, and $\mathbf{x} = (\mathbf{y}, \mathbf{z})$ be the augmented or "complete" data vector. Denote the p.d.f for $\mathbf{x}$ by $p_c(\boldsymbol{x};\boldsymbol{\Psi})$, and the complete data likelihood function by $L_c(\boldsymbol{\Psi}) = p_c(\boldsymbol{x};\boldsymbol{\Psi})$.

It is in cases in which it is easier to find the maximum likelihood estimate for $L_c(\boldsymbol{\Psi})$ than it is for $L(\boldsymbol{\Psi})$ that the EM algorithm is appropriate. However, since the "missing data" is only a convenient fiction that of course cannot be observed, one cannot simply maximize $L_c(\boldsymbol{\Psi})$. Intuitively, one can think of the EM algorithm as solving the problem of the "missing data" by "filling in" these values with their expectation with respect to the observed data and a current estimate of the parameters. One then proceeds iteratively, re-estimating the parameters and "filling in" the data.

To be precise, in the expectation (E-step) of the EM algorithm we form a k^{th} estimating function for the likelihood by calculating an expectation as follows:

$$Q^{(k)}(\boldsymbol{\Psi}) = E_{\widehat{\boldsymbol{\Psi}}^{(k)}}[logL_c(\boldsymbol{\Psi})|\mathbf{y}] = \int p_c(\boldsymbol{y} = \boldsymbol{d}, \boldsymbol{z};\boldsymbol{\Psi})p(\boldsymbol{z}|\boldsymbol{y} = \boldsymbol{d};\widehat{\boldsymbol{\Psi}}^{(k)})d\boldsymbol{z} \tag{2.4}$$

In the maximization (M-step) of the EM algorithm, we then find a value $\widehat{\boldsymbol{\Psi}}^{(k+1)}$ that maximizes the function $Q^{(k)}(\boldsymbol{\Psi})$. Since the EM algorithm is intended to be applied to problems for which the maximization of the complete data likelihood is computationally tractable, this step is generally straight-forward. For many problems, closed form solutions may be used. However, in other cases one may have to resort to numerical methods.

The E- and M-steps are repeated in turn until some stopping criteria is met. A typical criteria is that the difference $L(\widehat{\boldsymbol{\Psi}}^{(k+1)}) - L(\widehat{\boldsymbol{\Psi}}^{(k)})$ changes by only a small amount. See figure 2-1 for pseudo-code for the algorithm.

The crucial feature of the EM algorithm is that the "incomplete data" likelihood function $L(\boldsymbol{\Psi})$ is monotonically non-decreasing after each iteration in most reasonable situations (see [35] for a proof of this). That is, it can be shown that:

$$L(\widehat{\boldsymbol{\Psi}}^{(k+1)}) \geq L(\widehat{\boldsymbol{\Psi}}^{(k)})$$

Hence, if the "incomplete data" $L(\boldsymbol{\Psi})$ likelihood function is bounded, the EM algorithm will converge. In general, convergence can be slow and a global maximum is by no means assured.

However, for estimating the parameters of linear mixed-effects models when a sufficient amount of data has been collected, these problems are usually not significant [30, 27].

Note that if the complete data log likelihood function $\log L_c(\mathbf{\Psi})$ is linear in the "missing data," one can generate $Q^{(k)}(\mathbf{\Psi})$ in a particularly straight-forward manner. One simply calculates the expected value of the "missing data" $\mathbf{z}$ conditioned on the observed data (and using the $\widehat{\mathbf{\Psi}}^{(k)}$ parameter estimate):

$$\widehat{\mathbf{z}}^{(k)} = E_{\widehat{\mathbf{\Psi}}^{(k)}}[\mathbf{z}|\mathbf{y}] = \int \boldsymbol{z} p(\boldsymbol{z}|\boldsymbol{y} = \boldsymbol{d}; \widehat{\mathbf{\Psi}}^{(k)}) d\boldsymbol{z}$$

This estimate of the "missing data" is then "plugged in" to $L_c(\mathbf{\Psi})$ along with the observed data. However, in cases in which the complete data log likelihood function is not linear in the "missing data," one will need to compute higher moments and possibly interaction terms between the random variables. The linear mixed-effects model as discussion in section 2.1 is such a case; Appendix C.2 provides more complete details on how to estimate the model parameters using the EM algorithm.

EM Algorithm
- $k \leftarrow 0$
- $\mathbf{\Psi}^{(0)} \leftarrow$ initial guess
- do
  - E-step: construct likelihood approximation $Q^{(k)}(\mathbf{\Psi}) = E_{\widehat{\mathbf{\Psi}}^{(k)}}[logL_c(\mathbf{\Psi})|\mathbf{y}]$
  - M-step: $\widehat{\mathbf{\Psi}}^{(k+1)} \leftarrow \arg\max_{\mathbf{\Psi}}\{Q^{(k)}(\mathbf{\Psi})\}$
  - $k \leftarrow k + 1$
- while $L(\widehat{\mathbf{\Psi}}^{(k+1)}) - L(\widehat{\mathbf{\Psi}}^{(k)}) > \epsilon$

Figure 2-1: Pseudo-code for the Expectation-Maximization (EM) algorithm, a general method for iteratively computing maximum likelihood (ML) estimates. Here, $\mathbf{\Psi}$ is a vector of parameters to be estimated, $L(\mathbf{\Psi})$ is the data likelihood function, $L_c(\mathbf{\Psi})$ is the complete data likelihood function, and ϵ is a small numerical tolerance threshold. The algorithm proceeds by forming an estimate of the data likelihood function, by taking the conditional expectation of the complete data likelihood function with respect to the observed data and the current estimates of the parameter values (E-step). In the M-step, new parameter estimates are obtained by maximizing the likelihood of the estimated data likelihood function obtained in the previous E-step.

2.3 Splines

Splines are piecewise polynomials with boundary continuity and smoothness constraints. They are widely used in fields such as computer-aided design, image processing and statistics [7, 56, 16, 27]. A typical application involves using splines to generate a smooth curve that interpolates or approximates the shape of a discrete set of data points. The use of piecewise low-degree polynomials helps to avoid problems of overfitting, numerical instability, and oscillations that arise if single high-degree polynomials were used.

Although higher order splines are useful in some applications, in this thesis only cubic splines will be used and so the subsequent discussion will be restricted to this case. Cubic

splines have a number of desirable properties. For instance, cubic polynomials are the lowest degree polynomials that allow for a point of inflection. It can also be shown that cubic splines are a good mathematical model for a physical system consisting of a thin elastic beam, which produces "fair" curves when bent into different shapes [48].

A more formal justification for using cubic splines is as follows. Suppose observations are made at m time points $t_1 \dots t_m$ giving a vector $\boldsymbol{d} = (d_1, \dots, d_m)^T$ of data points. The typical parametric regression problem in statistics involves finding parameters $\widehat{\boldsymbol{\theta}}$ for a function $y(t; \boldsymbol{\theta})$ that minimize the sum of squared error terms:

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \{\sum_{i=1}^{m} [d_i - y(t_i; \boldsymbol{\theta})]^2\} \tag{2.5}$$

Now, suppose that instead of seeking parameters for a fixed function $y(t; \boldsymbol{\theta})$, we perform the minimization in equation 2.5 over all possible functions from a set $\mathcal{F}$ of twice differentiable functions. The solution would not be unique, since any function that interpolates the data exactly would yield zero error. Further, there is nothing to prevent these functions from oscillating wildly around the data points. One approach to this problem is to introduce a penalty for the "roughness" or local variation of a function. The penalized regression problem can then be formulated as:

$$\widehat{y}(t) = \arg\min_{y \in \mathcal{F}} \{\sum_{i=1}^{m} [d_i - y(t_i)]^2 + \lambda \int_{t_1}^{t_m} [y''(t)]^2 dt\} \tag{2.6}$$

Here, $\lambda > 0$ is a pre-specified value called the smoothing parameter, and the integral of the squared second derivative of the function acts as a penalty on its local variation. It can be proved that equation 2.6 has a unique solution, which is a cubic spline (see [16] for details). A key to this proof is the fact that cubic splines exhibit the *minimal curvature property*. This means that if $y(t)$ is a cubic spline, for any other $f(t) \in \mathcal{F}$ such that $y(t_i) = f(t_i)$, then $\int_{t_1}^{t_m} [y''(t)]^2 \leq \int_{t_1}^{t_m} [f''(t)]^2$. Thus, in some sense cubic splines are the "smoothest" curves capable of interpolating the data.

A cubic spline can be represented with the following equation:

$$y(t) = \sum_{i=1}^{q} c_i s_i(t) \qquad t_{min} \leq t < t_{max} \tag{2.7}$$

Here, t is a parameter (e.g., time), $s_i(t)$ are polynomials, and c_i are the coefficients. The typical way to represent a piecewise cubic curve is simply:

$$s_{4j+l}(t) = \begin{cases} t^{l-1}, & x_j \leq t \leq x_{j+1} \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

Here, $l = 1 \dots 4$, $j = 0 \dots q/4 - 1$ and the x_j's denote the *break-points* of the piecewise polynomials. Thus, we have $q/4$ cubic polynomials that can be denoted $p_j(t) = \sum_{l=1}^{4} c_{4j+l} t^{l-1}$. In order to determine the coefficients of these polynomials, q equations are required. If one specifies a value d_j plus continuity constraints up to the second derivative for the piecewise polynomial at each break-point x_j for $j = 1 \dots q/4 - 1$, four equations are obtained for each of the $q/4 - 1$ internal break-points:

$$p_j(x_j) = d_j$$

$$p_j(x_j) = p_{j-1}(x_j)$$
$$p'_j(x_j) = p'_{j-1}(x_j)$$
$$p''_j(x_j) = p''_{j-1}(x_j)$$

Additionally, specifying values for the end-points $p_0(x_0) = d_0$ and $p_{q/4-1}(x_{q/4}) = d_{q/4}$ yields a total of $q-2$ equations. Thus, in order to solve for the spline coefficients, an additional two equations are needed. Typically, these equations are obtained by specifying the first or second derivatives at the two end-points. Note that since one explicitly specifies the values $p_j(x_j)$ at the break-points, the formulation in equation 2.8 is particularly useful for defining *interpolating* splines.

2.3.1 B-splines

While the method discussed so far for defining cubic splines is easy to understand, it is not the most flexible or mathematically convenient formulation for many applications. Alternately, one can write a cubic polynomial $p(t) = a_3t^3 + a_2t^2 + a_1t + a_0$ in terms of a normalized set of four normalized *basis* functions:

$$p(t) = c_3b_3(t) + c_2b_2(t) + c_1b_1(t) + c_0b_0(t)$$

Here, each $b_i(t)$ is also a cubic polynomial and $\sum_{i=0}^{3} b_i(t) = 1$ for any t. In fact, there is no unique representation for a given cubic polynomial, since it can be written in terms of an arbitrary basis.

A very popular basis is the B-spline basis, which has a number of desirable properties. The texts by Rogers and Adams [48] and Bartels *et al* [7] give a full treatment of this topic. Once again, the discussion here will be limited to features relevant to this thesis. Most significantly for the application of fitting curves to gene expression time-series data, it is quite convenient with the B-spline basis to obtain approximating or *smoothing* splines rather than interpolating splines. Smoothing splines use fewer basis coefficients than there are observed data points, which is helpful in avoiding overfitting. In this regard, the basis coefficients c_i can be interpreted geometrically as *control points*, or the vertices of a polygon that control the shape of the spline but are not interpolated by the curve. It can be shown that the curve lies entirely within the convex hull of this controlling polygon. Further, each vertex exerts only a local influence on the curve, and by varying the vector of control points and another vector of knot points (discussed below), one can easily change continuity and other properties of the curve.

The normalized B-spline basis can be calculated using the Cox-deBoor recursion formula [48]:

$$b_{i,1}(t) = \begin{cases} 1, & x_i \le t < x_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

$$b_{i,k}(t) = \frac{(t - x_i)b_{i,k-1}(t)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - t)b_{i+1,k-1}(t)}{x_{i+k} - x_{i+1}} \tag{2.10}$$

Here, k is the order of the basis polynomials (degree $k-1$) and $2 \le k \le q$. As with any basis, $\sum_{i=1}^{q} b_{i,k}(t) = 1$ for any t. Each basis function is also non-negative for any t.

The values x_i are called *knots*, where $i = 1 \ldots q + k$. A *uniform* knot vector is one in which the entries are evenly spaced, i.e., $\boldsymbol{x} = (0, 1, 2, 3, 4, 5, 6, 7)^T$. If a uniform knot vector is used, the resulting B-spline is called periodic since the basis functions will be translates

of each other, i.e., $b_{i,k}(t) = b_{i-1,k}(t-1) = b_{i+1,k}(t+1)$. See figure 2-2 for an example. For a periodic cubic B-spline ($k = 4$), the equation specifying the curve can be written as:

$$y(t) = \sum_{i=1}^{q} c_i b_{i,4}(t) \quad \text{for} \quad x_4 \leq t \leq x_{q+1} \tag{2.11}$$

Notice that the spline is not defined over the entire range of knots x_j. The reason for this can be seen from inspecting the Cox-deBoor recursion formula in equations 2.9 and 2.10 and the spline equation 2.11: any point t on the spline curve $y(t)$ is always a combination of three basis function. Figure 2-2 shows this effect. In practice, this means that one must add three additional points (outside the range the spline will actually be defined on) to each end of the knot vector.

The B-spline basis allows one to write a particularly simple matrix equation when fitting splines to a set of data points. Suppose observations are made at m time points $t_1 \ldots t_m$ giving a vector $\boldsymbol{d} = (d_1, \ldots, d_m)^T$ of data points. We can then write the matrix equation $\boldsymbol{d} = \boldsymbol{S}\boldsymbol{c}$, where $\boldsymbol{c}$ is the vector of q control points and $\boldsymbol{S}$ is a m by q matrix where $[S]_{ij} = b_{j,4}(t_i)$. If $q = m$ (the number of control points equals the number of data points), then $\boldsymbol{S}$ is square and the equation may be solved by matrix inversion, yielding an interpolating spline. However, as discussed this may lead to overfitting and it is often desirable to use fewer control points than data points to obtain a smoothing or approximating spline. In this case, the matrix equation must be solved in the least-squares sense, which yields $\boldsymbol{c} = (\boldsymbol{S}^T\boldsymbol{S})^{-1}\boldsymbol{S}^T\boldsymbol{d}$.

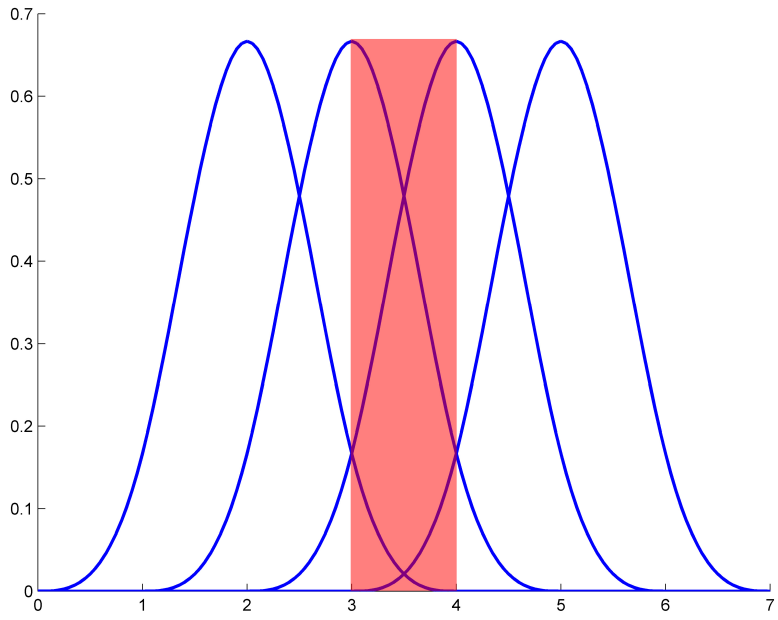

Figure 2-2: The B-spline basis functions are periodic (translates of one another) when a uniform knot vector is used. Shown here is the normalized cubic B-spline basis ($k = 4$) with knot vector $\boldsymbol{x} = (0, 1, 2, 3, 4, 5, 6, 7)^T$. Note that the B-spline will only be defined in the shaded region $3 \leq t \leq 4$, where all four basis functions overlap.

2.4 Gene Expression Time-Series Linear Mixed-Effects Spline Model

Now that the necessary background material has been presented, the model for representing gene expression time-series data as a set of continuous curves will be described. The core idea behind the model is that cubic B-splines are used to fit a gene's expression profile, while the spline control points are simultaneously constrained based on co-expressed genes' profiles. That is, splines' control points are modelled as linear combinations of random (gene specific) and fixed (shared by co-expressed genes) effects.

The model can be described formally as follows. Assume that DNA microarray time-series data has been collected for n_G genes where time-points are chosen from a set of n_T possible times denoted $T = \{t_1 \dots t_{n_T}\}$. Each gene i's expression values may be sampled at different time-points from T, and we will denote these n_{T_i} time-points by T_i where $T_i \subseteq T$. Let d_{it} be the expression measurement for the i^{th} gene at time-point $t \in T_i$ and denote the n_{T_i}-dimensional vector of such measurements by $\boldsymbol{d}_i$. We will call a set of co-expressed genes a *class*, and denote it by C_j where there are n_C classes each containing n_{C_j} genes. Cubic B-splines with q control points will be used to represent expression patterns. Let $\boldsymbol{s}(t)$ denote a q-dimensional vector of cubic B-spline basis functions defined on T, i.e., $\boldsymbol{s}(t) = (b_{1,4}(t), \dots, b_{q,4}(t))^T$. Let $\boldsymbol{S}_i$ denote a n_{T_i} by q matrix, in which each row consists of $[\boldsymbol{s}(t)]^T$ evaluated at each $t \in T_i$.

Let $\mathbf{y}_i$ denote the n_{T_i}-dimensional random vector of expression levels of gene i at the time-points $t \in T_i$. Denote by $\boldsymbol{\mu}_j$ a q-dimensional vector that represents a fixed effect or common mean of spline control points for genes in class j. Let $\boldsymbol{\gamma}_i$ be a q-dimensional random vector that represents the gene specific spline control points (random effect), for gene i in class j. Assume that $\boldsymbol{\gamma}_i$ is normally distributed with mean zero and a q by q covariance matrix $\boldsymbol{\Gamma}_j$. Finally, we will let $\boldsymbol{\epsilon}_i$ be an additive noise term, a n_{T_i}-dimensional normally distributed random variable with mean zero and variance $\sigma^2\boldsymbol{I}$. A linear mixed-effects model for a gene's expression values can then be written:

$$\mathbf{y}_i = \boldsymbol{S}_i(\boldsymbol{\mu}_j + \boldsymbol{\gamma}_i) + \boldsymbol{\epsilon}_i \tag{2.12}$$

For convenience of notation, let $\boldsymbol{\Psi} = (\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_{n_C}, \boldsymbol{\Gamma}_1, \dots \boldsymbol{\Gamma}_{n_C}, \sigma)^T$ denote a vector of all the parameters. Note that $\boldsymbol{\mu}_j$ and $\boldsymbol{\Gamma}_j$ are parameters common to genes in the same class, and σ is a noise variance parameter shared by all genes. The model presented here is almost identical to the general linear mixed-effects model presented in section 2.1, if we let $\boldsymbol{S}_i = \boldsymbol{A}_i = \boldsymbol{B}_i$ and consider the separate class means and covariance matrices.

Our objective is then to use the model described in this section to *predict* a spline curve based on the data. As discussed in section 2.1, the random effects predictor can be written as:

$$\widehat{\boldsymbol{\gamma}}_i = \boldsymbol{\Gamma}_j\boldsymbol{S}_i^T(\boldsymbol{S}_i\boldsymbol{\Gamma}_j\boldsymbol{S}_i^T + \sigma^2\boldsymbol{I})^{-1}(\boldsymbol{d}_i - \boldsymbol{S}_i\boldsymbol{\mu}_j)$$

We can then generate a predictor spline that can be evaluated at any time $t \in T$:

$$\widehat{y}_i(t) = [\boldsymbol{s}(t)]^T(\boldsymbol{\mu}_j + \widehat{\boldsymbol{\gamma}}_i) \tag{2.13}$$

The spline predictor can be expanded in a form that can yield some additional intuition (see Appendix C.3 for the derivation):

$$\widehat{y}_i(t) = [\boldsymbol{s}(t)]^T\{[\boldsymbol{I} - (\boldsymbol{S}_i^T\boldsymbol{S}_i + \sigma^2\boldsymbol{\Gamma}_j^{-1})^{-1}\boldsymbol{S}_i^T\boldsymbol{S}_i]\boldsymbol{\mu}_j + (\boldsymbol{S}_i^T\boldsymbol{S}_i + \sigma^2\boldsymbol{\Gamma}_j^{-1})^{-1}\boldsymbol{S}_i^T\boldsymbol{d}_i\} \tag{2.14}$$

Equation 2.14 suggests the affect on the gene's predicted spline curve of varying the number of observed time-points n_{T_i}, the noise variance σ, and the covariance $\mathbf{\Gamma}_j$ of spline control points of genes in the same class.

Consider the case in which $\mathbf{\Gamma}_j$ is large (i.e., in the sense of the trace of the matrix), indicating that the spline control points of genes in the same class are not very similar. Further, suppose that σ is small, implying that there is low additive noise. Finally, assume that n_{T_i} is large, which means that $\boldsymbol{S}_i^T \boldsymbol{S}_i$ will be large since the B-spline basis functions are always non-negative. Thus, $(\boldsymbol{S}_i^T \boldsymbol{S}_i + \sigma^2 \mathbf{\Gamma}_j^{-1})^{-1} \approx (\boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1}$. Substituting this approximation into equation 2.14 gives us $\widehat{y}_i(t) \approx [\boldsymbol{s}(t)]^T \boldsymbol{S}_i^{-1} \boldsymbol{d}_i$. The interpretation of this is that if there is a lot of data, the noise is low, and genes in the same class do not have very similar spline control points, then the estimate of a gene's spline curve will be based largely on the data.

Consider the alternate case in which $\mathbf{\Gamma}_j$ is small, σ is large, and n_{T_i} is small, which gives us:

$$\widehat{y}_i(t) \approx [\boldsymbol{s}(t)]^T [(\boldsymbol{I} - \mathbf{\Gamma}_j \boldsymbol{S}_i^T \boldsymbol{S}_i / \sigma^2) \boldsymbol{\mu}_j + \mathbf{\Gamma}_j \boldsymbol{S}_i^T \boldsymbol{d}_i / \sigma^2] \approx [\boldsymbol{s}(t)]^T \boldsymbol{\mu}_j$$

Thus, if there is little data, high noise, and the genes in the same class have similar spline control points, the class mean $\boldsymbol{\mu}_j$ will be what is mostly used in predicting a gene's spline curve.

In order to evaluate the predictor given in equation 2.13, the unknown parameters of the model must be estimated. This can be accomplished using the EM algorithm as described in section 2.2. The only difference is that now we must estimate parameters $\boldsymbol{\mu}_j$ and $\mathbf{\Gamma}_j$ for each class C_j. Note, however, that the estimations cannot be performed separately, as the classes are tied to each other through the common σ parameter. See Appendix C.2.1 for details on the solution.

The time-complexity of each iteration of the EM algorithm for estimating the parameters can be derived as follows. For each of the n_G genes, the $\widehat{\boldsymbol{\gamma}}_i$ predictor must be calculated. This is an $O(q^3)$ calculation (recall that q is the number of spline control points), since it involves matrix multiplications and inversions of q by q matrices[2]. Calculating the parameters for the n_C classes will also require $O(q^3)$ calculations. Thus, the running time of each iteration of the EM algorithm is $O((n_C + n_G)q^3)$. In practice, since $n_G \gg n_C$ and q is small, the running time is essentially linear in the number of genes n_G.

[2]There are matrix inversion and multiplication algorithms that run faster than $O(q^3)$, but these are usually only practically faster if q is large. Since q is relatively small in this problem, the $O(q^3)$ assumption is reasonable.

Chapter 3

Aligning Continuous Gene Expression Time-Series Profiles

In this chapter, a framework for aligning continuous gene expression time-series profiles will be presented. The core idea behind this method is to search for a function that transforms the time-scale of one gene expression process into that of another, while minimizing a measure of error between expression profiles in the two processes. A number of error measures are possible, but in general measures that operate on sets of genes will be more robust to noise and other problems inherent in DNA microarray data. Thus, in order to apply the method outlined below, a set of genes must first be chosen whose members are expected to show the same general temporal alterations across the two processes being studied. In some cases this set might include all expressed genes, although in other circumstances one might choose a subset of genes known to be participating in a particular process (e.g., cell cycle regulated genes).

An error measure that operates on sets of genes only over the overlapping parts of expression curves will then be introduced. An alternate measure will also be described that is useful in some cases for comparing wild-type with knock-out expression data. Next, the algorithm used to search for minimizing parameters will be discussed. Finally, some methods will be introduced for speeding up the alignment algorithm on large data sets.

3.1 Alignment Framework

The alignment method can be formalized as follows. Suppose a time-series of gene expression data has been collected for a set C of n genes throughout two experiments. The first experiment shall be referred to as the *reference* experiment, and the second shall be called the *warped* experiment for reasons soon to become obvious. Let $\widehat{y}_i^j(t)$ denote the predicted spline for gene i's expression levels in experiment j, where $t \in T^j = [t^j_{min}, t^j_{max}]$ for $j \in \{1, 2\}$ and $i = 1 \ldots n$.

Let $h(t; \boldsymbol{\theta}) : T^1 \mapsto T^2$ be a parameterized *time-warping* function with parameter vector $\boldsymbol{\theta}$. Further, assume that $h(t; \boldsymbol{\theta})$ is monotonically increasing and hence invertible. Thus, $h(t; \boldsymbol{\theta})$ is a function that maps a time-point in the first experiment to a point in the second (and $h^{-1}(t; \boldsymbol{\theta})$ does the opposite). One choice is a linear time-warping function, which can be written as:

$$h(t; \boldsymbol{\theta}) = (t - \theta_2)/\theta_1 \tag{3.1}$$

Here, θ_2 is the *translation* parameter, and θ_1 is the *stretch/squash* parameter. The linear time-warping function will be used in the applications portion of this thesis (Chapter 4).

In general, our goal will be to find parameters $\widehat{\boldsymbol{\theta}}$ for the warping function $h(t;\boldsymbol{\theta})$ that minimize an error function over the set of genes C. We will also usually want to constrain the parameter choices in some way. For instance, for the linear time-warping function in equation 3.1, we must have $\theta_1 > 0$ to avoid a "time-reversal" effect. The constrained minimization problem can be stated generally as:

$$\widehat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}\in\Omega}\{E_C(\boldsymbol{\theta})\} \tag{3.2}$$

Here, Ω denotes the set of possible parameter values (i.e., those that satisfy specified constraints). This set is often constructed implicitly, by specifying m (non)-linear inequality constraints of the form $g_k(\boldsymbol{\theta}) \leq 0$, where $k = 1 \ldots m$. $E_C(\boldsymbol{\theta})$ in equation 3.2 is an error measure over the set of genes C. A particular choice for this error measure will be described in the following section.

3.2 Alignment Error Measure

It is natural to begin constructing the alignment error measure $E_C(\boldsymbol{\theta})$ by considering the squared error between values of the predicted splines for genes in the reference and warped experiments:

$$\delta_i^2(t;\boldsymbol{\theta}) = [\widehat{y}_i^1(t) - \widehat{y}_i^2(h(t;\boldsymbol{\theta}))]^2$$

So far we have a point-wise error measure, and we'd like a global measure that somehow "averages" the alignment error over the entirety of the expression curves. In practice, though, the biological processes being studied in two experiments to be aligned can be expected to overlap, but not necessarily to begin and end during the same parts of the process. Thus, a measure that is "global" but only over the overlapping parts of the expression curves will be introduced.

Denote the starting point of the overlap by $\alpha(\boldsymbol{\theta}) = \max\{t_{min}^1, h^{-1}(t_{min}^2;\boldsymbol{\theta})\}$ and the ending point by $\beta(\boldsymbol{\theta}) = \min\{t_{max}^1, h^{-1}(t_{max}^2;\boldsymbol{\theta})\}$. We then define the alignment error $e_i^2(\boldsymbol{\theta})$ for each gene as:

$$e_i^2(\boldsymbol{\theta}) = \frac{\int_{\alpha(\boldsymbol{\theta})}^{\beta(\boldsymbol{\theta})} \delta_i^2(t;\boldsymbol{\theta})dt}{\beta(\boldsymbol{\theta}) - \alpha(\boldsymbol{\theta})} \tag{3.3}$$

In words, the alignment error for each gene is equal to the squared distance between the gene's curve in the reference experiment and that in the warped experiment, averaged over the duration of overlap between the two curves. Note that the term in the denominator of equation 3.3, $\beta(\boldsymbol{\theta}) - \alpha(\boldsymbol{\theta})$, is the length of the interval of overlap. This term effectively penalizes small overlaps, and prevents the warped curve from "shifting away" or "shrinking" excessively.

Our error measure is still not complete in general, since as discussed it is best to minimize the error for a set of genes. So, we define the complete measure as a weighted sum of the individual errors:

$$E_C(\boldsymbol{\theta}) = \sum_{i=1}^{n} w_i e_i^2(\boldsymbol{\theta}) \tag{3.4}$$

The w_i's are weighting coefficients that sum to one. In many applications, uniform weighting with $w_i = 1/n$ is most appropriate. However, in other cases we may want to use an alternate

weighting such as that described in the sub-section below.

3.2.1 Alternate Weighting for the Error Measure for Knockout Experiments

Use of non-uniform weights in equation 3.4 may be appropriate in the following circumstance. In some knockout experiments, the majority of genes' expression patterns would not be expected to change as compared to the wildtype case. However, a subset of genes may be highly affected. In this case, we may want to down-weight the contribution of the highly affected genes, since their knockout and wildtype expression profiles would not be expected to align well. One way of achieving this is to require that the product $w_i e_i^2(\boldsymbol{\theta})$ be the same for all genes, i.e., the weight will be inversely proportional to the error. This assumption then allows us to derive an alternate expression for the error $E_C(\boldsymbol{\theta})$:

$$\begin{aligned} w_i e_i^2(\boldsymbol{\theta}) = K \Rightarrow \\ E_C(\boldsymbol{\theta}) = \sum_{i=1}^{n} w_i e_i^2(\boldsymbol{\theta}) = nK \\ w_i = \frac{K}{e_i^2(\boldsymbol{\theta})} \Rightarrow \\ \sum_{i=1}^{n} w_i = 1 = \sum_{i=1}^{n} \frac{K}{e_i^2(\boldsymbol{\theta})} \Rightarrow \\ K = \frac{1}{\sum_{i=1}^{n} 1/e_i^2(\boldsymbol{\theta})} \Rightarrow \\ E_C(\boldsymbol{\theta}) = \frac{n}{\sum_{i=1}^{n} 1/e_i^2(\boldsymbol{\theta})} \end{aligned}$$

As before, the objective is to minimize $E_C(\boldsymbol{\theta})$, or in this case equivalently to maximize $\sum_{i=1}^{n} 1/e_i^2(\boldsymbol{\theta})$.

3.3 Minimizing the Alignment Error Measure

The problem of finding the parameters $\widehat{\boldsymbol{\theta}}$ that minimize $E_C(\boldsymbol{\theta})$ subject to constraints as described in equation 3.2, is in general a constrained non-linear optimization problem. Even for the case of a linear time-warping function $h(t;\boldsymbol{\theta})$ as in equation 3.1, the error function $E_C(\boldsymbol{\theta})$ will be a seventh degree polynomial in the parameters. Additionally, in this case the constraints $\alpha(\boldsymbol{\theta}) - \beta(\boldsymbol{\theta}) < 0$ and $-\theta_1 < 0$ are necessary.

Since closed form solutions for the constrained non-linear optimization problem described are not possible, numerical methods must be used. For the applications presented in this thesis, both the Matlab Optimization Toolbox [33] and the publicly available package CFSQP (C code for Feasible Sequential Quadratic Programming) [31] were found to perform well. A full discussion of the numerical optimization methods used by these programs is beyond the scope of this thesis. Thus, only the key conceptual points will be presented. Also, only CFSQP will be discussed; the Matlab Optimization routines are similar in principle.

The basic idea behind the CFSQP algorithm is to take steps toward a minimizing solution for equation 3.2, by iteratively solving the easier problem of finding a minimum of a quadratic approximation of the objective function. This approach is called Sequential Quadratic Programming (SQP), which is essentially a generalization of Newton's method

and can achieve super-linear convergence. CFSQP additionally ensures that feasible iterates (i.e., those satisfying the constraints) are generated during the search.

A high-level overview of the steps of the CFSQP algorithm are as follows. First, the Lagrangian is formed (i.e., the constraints are incorporated with Lagrange multipliers):

$$L(\boldsymbol{\theta}, \boldsymbol{\lambda}) = E_C(\boldsymbol{\theta}) + \sum_{j=1}^{m} \lambda_j g_j(\boldsymbol{\theta})$$

Next, the objective function is replaced by a quadratic approximation:

$$Q(\boldsymbol{d}) = \nabla E_C(\boldsymbol{\theta})^T \boldsymbol{d} + \frac{1}{2} \boldsymbol{d}^T \widetilde{\boldsymbol{H}}(\boldsymbol{\theta}^*, \boldsymbol{\lambda}^*) \boldsymbol{d}$$

Here, $\widetilde{\boldsymbol{H}}(\boldsymbol{\theta}^*, \boldsymbol{\lambda}^*)$ is a positive definite approximation of the Hessian (matrix of second derivatives) of $L(\boldsymbol{\theta}, \boldsymbol{\lambda})$ evaluated at a feasible point $(\boldsymbol{\theta}^*, \boldsymbol{\lambda}^*)$. This approximation method is similar to using the lower order terms from a Taylor expansion to form a local estimate of a function. The constraint functions $g_j(\boldsymbol{\theta})$ are then replaced with linear approximations:

$$\widetilde{g}_j(\boldsymbol{d}) = g_j(\boldsymbol{\theta}^*) + \nabla g_j(\boldsymbol{\theta}^*)^T \boldsymbol{d}$$

The problem then becomes one of solving a quadratic program:

$$\boldsymbol{d}_0 = \arg \min_{\boldsymbol{d}} \{Q(\boldsymbol{d})\}$$

Where the minimization problem is subject to the constraints $\widetilde{g}_j(\boldsymbol{d}) \leq 0$ for $j = 1 \ldots m$. Note that since $\widetilde{\boldsymbol{H}}(\boldsymbol{\theta}^*, \boldsymbol{\lambda}^*)$ is a positive definite matrix, $Q(\boldsymbol{d})$ will be convex and thus a minimum point must exist. The vector $\boldsymbol{d}_0$ gives a direction of greatest descent for our approximation $Q(\boldsymbol{d})$ of the objective function $E_C(\boldsymbol{\theta})$. Using this direction, one could then perform a *line search* along $\boldsymbol{d}_0$, i.e., minimize the one-dimensional function $f(\rho) = E_C((1 - \rho)\boldsymbol{d}_0 + \boldsymbol{\theta}^* \rho)$. Two complications arise. First, moving along $\boldsymbol{d}_0$ may go outside the feasible reason. Second, moving in the direction $\boldsymbol{d}_0$ may not guarantee super-linear convergence. The CFSQP algorithm addresses both problems by computing a new descent direction that is a combination of $\boldsymbol{d}_0$, a strictly feasible direction, and a second-order correction term. This direction ensures local feasible iterates and under certain fairly general conditions, super-linear convergence can be guaranteed. See [31, 40] for more complete details.

3.4 Approaches for Speeding Up the Alignment Algorithm

As the use of a numerical optimization method, such as that described in section 3.3, does not guarantee convergence to a global minimum, multiple re-starts with different initial parameter settings may be necessary. Since the number of genes being aligned can be large, a stochastic version of the alignment algorithm can be used to reduce the computation time. The algorithm begins by choosing a random subset of n_R genes from the total set C of genes and random initial settings for the warping parameters from a uniform distribution. The minimization procedure as described in section 3.3 is then carried out and this process is repeated a fixed number of times, r, with a new random choice of initial warping parameters each time. Upon termination, the alignment parameters that corresponded to the lowest error are chosen. These parameters are then used as the starting conditions for the error

minimization procedure using the full set of genes. Figure 3-1 provides pseudo-code for this algorithm. See section 4.3 for empirical results of how this improvement reduced the running time without affecting convergence of the parameter values.

It is also possible to speed up the calculation of $e_i^2(\boldsymbol{\theta})$ in the case in which $h(\boldsymbol{\theta}, t)$ is linear (see equation 3.1). In this case, for a fixed $\boldsymbol{\theta}$ the integral in $e_i^2(\boldsymbol{\theta})$ can be solved in closed form. This involves finding all the spline segments that overlap. To be more precise, let $\boldsymbol{x}^j$ denote a q-dimensional vector of break-points, and denote by $f_k^j(t)$ the piecewise polynomial defined on $[x_k^j, x_{k+1}^j]$ for the predictor spline in experiment j. Then the alignment error term for a gene can be evaluated as:

$$e^2(\boldsymbol{\theta}) = \frac{1}{\beta(\boldsymbol{\theta}) - \alpha(\boldsymbol{\theta})} \{ \sum_{k=1}^{q} \int_{x_k^{1-}(\boldsymbol{\theta})}^{x_{k+1}^{1+}(\boldsymbol{\theta})} [f_k^1(t)]^2 dt + \sum_{k=1}^{q} \int_{x_k^{\prime 2-}(\boldsymbol{\theta})}^{x_{k+1}^{\prime 2+}(\boldsymbol{\theta})} [f_k^2(h(t;\boldsymbol{\theta}))]^2 dt \\ -2 \sum_{k=1}^{q} \sum_{l=1}^{q} \int_{z_{kl}^-(\boldsymbol{\theta})}^{z_{kl}^+(\boldsymbol{\theta})} f_k^1(t) f_l^2(h(t;\boldsymbol{\theta})) dt \} \tag{3.5}$$

The limits of integration are defined as follows:

$$\begin{aligned} x_k^{1-}(\boldsymbol{\theta}) &= \max\{x_k^1, \alpha(\boldsymbol{\theta})\} \\ x_k^{1+}(\boldsymbol{\theta}) &= \min\{x_k^1, \beta(\boldsymbol{\theta})\} \\ x_k^{\prime 2-}(\boldsymbol{\theta}) &= \max\{h^{-1}(x_k^2;\boldsymbol{\theta})), \alpha(\boldsymbol{\theta})\} \\ x_k^{\prime 2+}(\boldsymbol{\theta}) &= \min\{h^{-1}(x_k^2;\boldsymbol{\theta})), \beta(\boldsymbol{\theta})\} \\ z_{kl}^-(\boldsymbol{\theta}) &= \max\{x_k^{1-}(\boldsymbol{\theta}), x_l^{\prime 2-}(\boldsymbol{\theta})\} \\ z_{kl}^+(\boldsymbol{\theta}) &= \min\{x_k^{1+}(\boldsymbol{\theta}), x_l^{\prime 2+}(\boldsymbol{\theta})\} \end{aligned}$$

In equation 3.5, the convention is adopted that an integral is zero if its lower limit is greater than its upper limit. Note that since most of the spline segments will not overlap, most of the terms in the double summation will be zero. Since each term to be integrated is a polynomial in t, the entire integral can be solved in closed form. Note that this is true for fixed $\boldsymbol{\theta}$, since the overlap of the spline segments depends on $\boldsymbol{\theta}$.

If the stochastic version of the alignment algorithm is used, $h(t;\boldsymbol{\theta})$ is linear in $\boldsymbol{\theta}$, and the closed form solution in equation 3.5 is used, the alignment algorithm has running time $O(klq^2(rn_R+n)$. Here, k is the running time for each iteration of the non-linear optimization routine, and l is the number of iterations necessary for convergence of the optimization routine. Each iteration of this routine must compute the error function, which requires $O(q^2)$ computations using equation 3.5, where q is the number of spline control points used for each gene's curve. Recall that r denotes the number of random re-starts used in the stochastic version of the alignment algorithm, n_R is the number of genes picked randomly in each iteration, and n is the total number of genes being aligned.

Stochastic version of the alignment algorithm

for $i \leftarrow 1$ to r

$\boldsymbol{\theta}^{(i)} \leftarrow$ randomly selected parameters

$C_i \leftarrow$ a randomly selected subset of n_R genes from the set C

$\widehat{\boldsymbol{\theta}}^{(i)} \leftarrow \texttt{nonlinop}(E_{C_i}(\boldsymbol{\theta}), \boldsymbol{\theta}^{(i)})$

$j \leftarrow \arg\min_{i=1,\ldots,r}\{E_{C_i}(\widehat{\boldsymbol{\theta}}^{(i)})\}$

$\widehat{\boldsymbol{\theta}}^{(r+1)} \leftarrow \texttt{nonlinop}(E_C(\boldsymbol{\theta}), \widehat{\boldsymbol{\theta}}^{(j)})$

Figure 3-1: Pseudo-code for the stochastic version of the alignment algorithm, which is useful for reducing the running time of the alignment algorithm when a large number of genes are being processed. Here, r is a pre-specified number of random re-starts, $\boldsymbol{\theta}$ is a warping parameter vector, $E_C(\boldsymbol{\theta})$ is an alignment error function over a set of genes C, and $\texttt{nonlinop}(\cdot, \cdot)$ is a numerical constrained non-linear optimization routine, where the second argument is the starting "guess" for the parameter vector. The algorithm begins by choosing a random subset of n_R genes from the total set C of genes and random initial settings for the warping parameters from a uniform distribution. The minimization procedure as described in section 3.3 is then carried out and this process is repeated a fixed number of times, r, with a new random choice of initial warping parameters each time. Upon termination, the alignment parameters that corresponded to the lowest error are chosen. These parameters are then used as the starting values for the error minimization procedure using the full set of genes.

Chapter 4

Applications

In this section three applications of the continuous representation/alignment algorithms to DNA microarray expression data are presented. In the first application, it is shown that the linear mixed-effects spline model can effectively predict missing data values, achieving lower error for cases of one or two consecutive missing values than the best method previously reported in the literature. Second, it is demonstrated that the alignment algorithm can produce stable low-error alignments, allowing data from three different yeast cell cycle experiments (with period and phase differences) to be combined. Finally, it is shown that the alignment algorithm can be used to identify genes that behave differently in normal versus genetically modified cells.

4.1 Description of Data Sets Analyzed

The data sets analyzed consist of four publicly available DNA microarray time-series studies pertaining to the yeast *Saccharomyces cerevisiae* cell cycle [51, 60]. In these experiments, yeast cells were synchronized, which involved blocking progression of cells through the cycle and then releasing all cells from the block simultaneously. Several methods exist for synchronization, many involving the use of different carbon sources and incubation temperatures, which affect the growth rate of the yeast cells and hence the period of the cell cycles. Further, alternate synchronization methods arrest the cells during different phases of the cycle. Thus, while the data sets represent measurements of the same basic underlying biological process, the cell cycles observed can be expected to have different phases and periods.

Three of the data sets analyzed are from the work of Spellman *et al* [51], in which yeast cells were synchronized using three different methods and mRNA was collected for microarray analysis at various time-points. These three data sets will be identified by their synchronization method: alpha mating factor (alphaDS), cdc15 temperature sensitive mutant (cdc15DS), and cdc28 temperature sensitive mutant (cdc28DS). The sampling intervals and rates for these three experiments differ; see table 4.1 for a summary. The alphaDS and cdc15DS were obtained using printed cDNA microarrays. For these data sets, the control sample labelled with a second fluorescent dye consisted of time-series of gene expression data obtained from unsynchronized yeast cells. The cdc28DS was obtained using Affymetrix oligonucleotide microarrays.

Spellman *et al* processed the raw data as follows. For the data obtained using printed cDNA arrays, ratios of measurements from the two fluorescent channels were $\log_2$ trans-

formed, and normalized so that the mean of the series of values for each gene was equal to zero. The cdc28DS was converted to ratio form by dividing each expression value by the average of all expression time-point values for the gene. The ratios were then $\log_2$ transformed and normalized in the same manner as were the data sets obtained using printed cDNA microarrays.

Spellman *et al* then identified approximately 800 genes as cell cycle regulated and assigned these genes to five groups *G1*, *S*, *S/G2*, *G2/M*, and *M/G1*, corresponding to phases of the cell cycle. For this identification and assignment, they used an empirical method that relied on prior biological knowledge. First they calculated the Fourier transform of the time-series for each gene:

$$a_i^j = \frac{1}{m_j} \sum_{l=1}^{m_j} \sum_{k=1}^{n_{T_i}} \sin(\omega_l^j t_k + \phi_j) d_{ik} \tag{4.1}$$

$$b_i^j = \frac{1}{m_j} \sum_{l=1}^{m_j} \sum_{k=1}^{n_{T_i}} \cos(\omega_l^j t_k + \phi_j) d_{ik} \tag{4.2}$$

$$(\rho_i^j)^2 = (a_i^j)^2 + (b_i^j)^2 \tag{4.3}$$

Here, a_i^j and b_i^j are the coefficients for gene i in experiment j, which are obtained by averaging over a range of m_j periods ω_l^j, where $j \in \{\text{alpha}, \text{cdc15}, \text{cdc28}\}$ and $l = 1, \ldots, m_j$. The range of periods was determined empirically by observing the division time for the growing yeast cells in each experiment. The scalar d_{ik} is the $\log_2$ transformed expression value of gene i at the k^{th} measured time-point, and ϕ_j is the phase value for experiment j. An aggregate "cycling score" c_i was then calculated for each gene:

$$c_i = \sqrt{0.49(\rho_i^{\text{alpha}})^2 + 0.49(\rho_i^{\text{cdc15}})^2 + (\rho_i^{\text{cdc28}})^2} \tag{4.4}$$

The phase ϕ_j was set to zero for the alpha experiment, and for the other two experiments it was set to the value that maximized the sum of the $(\rho_i^j)^2$ scores for all genes.

A small number of genes known to be cycling from prior biological experiments was then selected. These genes were assigned to one of five phase classes, *G1*, *S*, *S/G2*, *G2/M*, or *M/G1*, based on biological knowledge about the phase of their peak activities. The average expression profile (concatenating all three experiments) was then calculated for the known genes in each phase class. A cut-off value for the cycling score c_i was then chosen, so that 91% of the known cycling genes exceeded this score. Unknown genes were classified as cycling if their scores exceeded this threshold.

For each unknown gene identified as cycling, correlation coefficients were computed with the gene's expression profiles (concatenating all three experiments) and the average expression profile of known genes in each of the five phase classes. An unknown gene was assigned to the phase class for which its expression profile exhibited the highest correlation. See Spellman *et al* [51] for complete details.

The fourth data set analyzed came from a study by Zhu *et al* [60], in which the genes fkh1 and fkh2 that encode transcriptional regulators were knocked out, and a time-series of expression values was measured in alpha mating factor synchronized cells. This data set was obtained using printed cDNA microarrays. The controls and data normalization method used were similar to those described above for the Spellman *et al* data sets.

data set	method of arrest	start	end	sampling
alphaDS	alpha mating factor	0m	119m	every 7m
cdc15DS	temperature sensitive cdc15 mutant	10m	290m	every 20m for 1 hr, every 10m for 3 hr, every 20m for final hr
cdc28DS	temperature sensitive cdc28 mutant	0m	160m	every 10m
Fkh1/2DS	alpha mating factor	0m	210m	every 15m until 165m, then after 45m

Table 4.1: Summary of the four publicly available gene expression time-series data sets analyzed in this thesis. The first three data sets are from the work of Spellman *et al* [51], in which yeast cells were synchronized using three different methods. The fourth data set is from a study by Zhu *et al* [60], in which the genes fkh1 and fkh2, encoding transcriptional regulators, were knocked out.

4.2 Prediction of Missing Values

The goal of this section is to demonstrate that the linear mixed-effects spline model presented in section 2.4 works well for real gene expression time-series data. While inspection of plots of the original data and the predicted splines show a "good" fit, an objective quality criteria is not obvious. Simply using the error between the predicted splines and the observed data is problematic, because such a measure would encourage overfitting. A second complication is that in the data sets analyzed, expression measurements were not replicated. So, it is not possible to estimate the experimental variance accurately.

Given these issues, the perspective taken in this thesis is a practical one: to demonstrate that the linear mixed-effects spline model can predict missing data points at least as well, or better than several other methods previously described in the DNA microarray analysis literature. The validation method can be described as follows. A gene's expression vector was picked at random and one, two, and three consecutive points were removed from the vector. Values for these points were then estimated using all remaining data (including that from all other genes). This process was repeated 100 times. A mean error was then computed for each case of one, two, or three consecutive missing data points:

$$\mathrm{err}(m) = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} |d_{g_i t_{ij}} - \widehat{s}'_{g_i}(t_{ij})| \tag{4.5}$$

Here, $n = 100$ is the number of trials, m is the number of removed values, g_i is the index of the gene selected in the i^{th} trial, t_{ij} is the j^{th} time-point removed in the i^{th} trial, and $\widehat{s}'_{g_i}(t)$ is the spline estimated for the gene selected in the i^{th} trial with m time-points removed. Note that if the selected gene was actually missing expression values at the m time-points to be removed, another gene was chosen.

In addition to the linear mixed-effects spline method presented in this thesis, three others previously described in the literature were used to estimate missing values: linear interpolation [1], spline interpolation using individual genes [11], and K-nearest neighbors (KNN) with $k = 20$ [53]. Results using the cdc15DS only are reported here, since this was the largest data set available and is thus expected to yield the most meaningful statistics[1].

[1]Qualitatively similar results were obtained using the other data sets.

Uniform B-splines with seven segments were used, as this produced the best results across all the data sets. Genes were assigned to five classes as described in section 4.1.

The linear mixed-effects spline method outperformed linear and standard spline interpolation in all cases, and was superior to K-nearest neighbors for the cases of one and two consecutive missing values. K-nearest neighbors performed slightly better in the case of three consecutive missing values, but this case is fairly extreme and probably not likely to occur often in practice. Figure 4-1 shows these results. Note that the significance of differences between the errors achieved by the various methods is unclear, since an estimate of the experimental variance in the data was not available. Thus, it is only meaningful to consider observed trends in the relative performances of the measures.

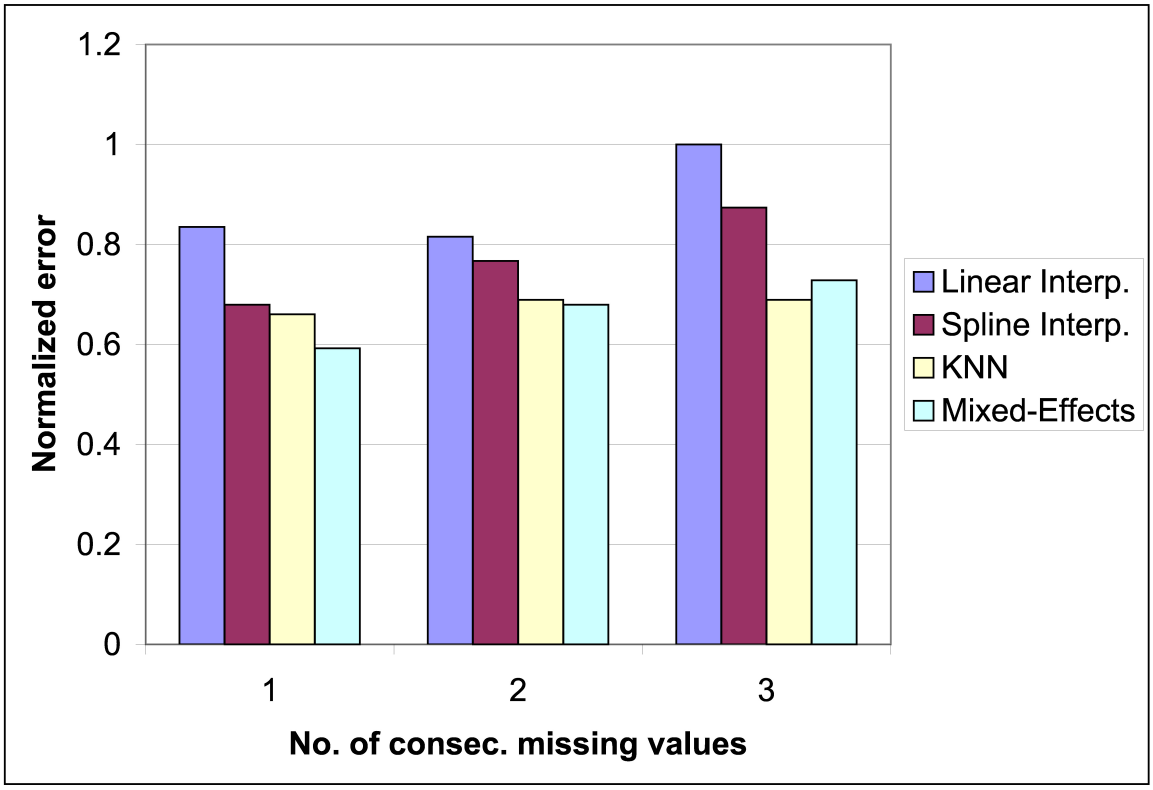

Figure 4-1: For predicting missing expression values, the linear mixed-effects spline method outperformed linear and standard spline interpolation in all cases, and was superior to K-nearest neighbors for the cases of one and two consecutive missing values. K-nearest neighbors (KNN) performed slightly better in the case of three consecutive missing values, but this case is fairly extreme and probably not likely to occur often in practice. The validation involved picking a gene at random from the data set and then removing one, two, and three consecutive time-points. Values for these time-points were then estimated using all remaining data and this process was repeated 100 times. A mean error was then computed for each case of one, two, or three consecutive missing data points. The errors shown were normalized by the largest error (linear interpolation with three consecutive missing values). The results shown were obtained using the cdc15DS.

4.3 Aligning Similar Biological Processes

In this section, the alignment algorithm is successfully applied to register three cell cycle data sets, cdc15DS, cdc28DS, and alphaDS, in which cells were synchronized using three different methods and thus exhibit different periods and phases. These data sets were also sampled at different rates and over different durations. As described in section 1.1, alignment of data sets can be generally useful for combining information from multiple experiments so that a sufficient amount of data is available for automated network discovery algorithms. Additionally, for the specific data sets analyzed here, the estimated alignment parameters may provide insights into the differences among the cell cycle synchronization methods.

The cdc15DS was used as a reference set and the alphaDS and cdc28DS were aligned against it using a linear warping $h(t;\boldsymbol{\theta}) = (t-\theta_2)/\theta_1$, and the full set of approximately 800 genes identified as cell cycle regulated as described in section 4.1. Uniform B-splines with seven segments were used for the reasons described in section 4.2.

For the cdc28DS, parameter estimates $\widehat{\theta}_1 = 1.42$ and $\widehat{\theta}_2 = 2.25$ were obtained with $E_C(\widehat{\boldsymbol{\theta}}) = 0.1850$. These results indicate that the cdc28DS cell cycle runs at approximately 1.4 times the speed of the cdc15DS cycle and starts approximately 5.5 minutes before[2]. For the alphaDS, parameter estimates $\widehat{\theta}_1 = 1.95$ and $\widehat{\theta}_2 = -5.89$ were obtained with $E_C(\widehat{\boldsymbol{\theta}}) = 0.1812$. These results indicate that the alphaDS cell-cycle runs at approximately 2 times the speed of the cdc15DS cycle and starts approximately 8 minutes before. Figure 4-2 provides a visual depiction of the cdc28DS to cdc15DS alignment.

Alignment for each data set took approximately 5.5 minutes on a 1 GHz Pentium III machine with 512 Mb of RAM, using the alignment algorithm with the improvements discussed in section 3.4; data set alignments took approximately 45 minutes without these improvements.

The estimated alignment parameters are biologically reasonable. Both the cdc15DS and cdc28DS were obtained using temperature sensitive mutant yeast, in which the cells are subjected to heat shock to achieve cell cycle arrest, and then placed at a lower temperature to resume the cycle. Such heat shock is known to reduce subsequent growth rates [51]. This is in contrast to the alphaDS, in which arrest was achieved through the addition of a chemical and no temperature shift occurred. Thus, the fact that the alphaDS has the shortest estimated period is biologically quite reasonable. The fact that the cdc28DS exhibits a somewhat shorter period than the cdc15DS may be explained by the fact that the subsequent release/growth temperature used for the cdc28DS was slightly higher, or that different strains of yeast were used. The fact that both the alphaDS and cdc28DS were estimated to begin before the cdc15DS is also reasonable. The synchronization method used for both the alphaDS and cdc28DS arrests cells during the G1 phase, whereas that used for the cdc15DS arrests cells in the M phase.

To validate the quality of these alignments two randomized analyses were performed: 1) alignments of the alphaDS to the cdc15DS with gene identity of expression profiles in the cdc15DS randomly permuted, and 2) alignments of the alphaDS to the cdc15DS using only subsets of randomly chosen genes. Only the alphaDS was used in these analyses; this data set was chosen because it is the smallest and presumably demonstrates worst-case results. The first analysis was designed to determine the quality of the alignment score $E_C(\widehat{\boldsymbol{\theta}})$. Essentially. we'd like to know the probability that the observed alignment

[2]This is obtained by calculating $h(10;\widehat{\boldsymbol{\theta}}) = 5.5$, since the cdc15DS starts at 10 minutes.

# genes	$\widehat{\theta}_1$	std $\widehat{\theta}_1$	$\widehat{\theta}_2$	std $\widehat{\theta}_2$
5	1.80	0.42	-7.70	16.41
10	1.89	0.21	-5.06	21.5
25	1.93	0.10	-4.99	7.07
50	1.93	0.13	-5.13	9.03
100	1.96	0.03	-6.86	2.42
200	1.95	0.02	-6.38	1.76
400	1.96	0.02	-6.12	1.30

Table 4.2: The estimated alignment parameters for the alphaDS to cdc15DS registration exhibit low variances even when fairly small sets of genes are used. Shown are the results of experiments in which random subsets of fixed size were sampled 100 times and alignment of the alphaDS to the cdc15DS was performed, using a linear warping function. The columns are as follows: number of genes used, stretch/squash parameter, standard deviation of this parameter, translation parameter, and standard deviation of this parameter. Note that the variance in the parameters decreases as more genes are used and there is convergence to the $\widehat{\theta}_1 = 1.95$ and $\widehat{\theta}_2 = -5.89$ settings found using the full set of genes.

score or lower could occur by chance, i.e., through alignment of random data sampled from the appropriate distribution. Since the underlying distribution of the expression data is difficult to model, a reasonable approach is to permute the available data. Rather than permuting the time-points, which would disrupt time-series dependencies and thus give poor alignments, the more stringent approach of permuting only the gene identities was used. The second analysis was designed to determine the sensitivity of the alignment parameters to the number of genes used.

For the first analysis, 200 trials were performed yielding $E_C(\widehat{\boldsymbol{\theta}})$ scores between 0.2562 and 0.3554, with 50% of the scores lying between 0.2780 and 0.3129. These results suggest that the actual alphaDS to cdc15DS $E_C(\widehat{\boldsymbol{\theta}})$ score of 0.1812 would not arise by chance.

For the second analysis, subsets of between 5 and 400 genes were sampled 100 times from the full set of cell cycle regulated genes (see table 4.2). As expected, the variance in the parameters decreases as more genes are used and there is convergence to the $\widehat{\theta}_1$ and $\widehat{\theta}_2$ estimates obtained using the full set of genes. Importantly, the variance in the parameters is low for even relatively small numbers of genes.

Thus, these analyses give evidence that the method presented in this thesis can be used to reliably align the three cell cycle data sets. These results compare favorably with those in Aach *et al* [1] that use the same data. In their case, they found that their actual alignment score was not at a low percentile when compared against alignments using randomized data (time-points permuted). Further, they indicate that poor results were obtained when small sets of genes were aligned (an analysis over a wide range of gene set sizes was not presented in their paper). The fact that the method presented in this thesis uses a continuous representation and fits only two parameters to all the genes, helps to explain its good performance on the cell cycle data. However, one must be careful in extrapolating conclusions from these results, since they are clearly dependent on the underlying data set.

4.4 Identifying Differentially Expressed Genes

In this section, the alignment method is used to help discover genes that are regulated by the Fkh2 yeast transcription factor. Fkh2 is a DNA binding protein known to be involved in regulating genes during the yeast cell cycle [50]. However, its precise role and which genes it regulates remain an open question. In a recent work, Simon *et al* [50] used a microarray based DNA-binding assay to identify genes in wildtype yeast that are bound by Fkh2. Unfortunately, binding of a protein to DNA near a gene in this assay provides only suggestive evidence that the protein regulates the gene's expression. One issue is that the large-scale binding assay used by Simon *et al* is known to produce many false-positives. Additionally, even if a protein does actually bind near a gene, the protein may be carrying out some function other than regulating that gene.

However, if DNA near a gene is bound by a protein *and* the gene's expression changes in an experiment in which the DNA-binding protein has been "knocked out," this is much stronger evidence for regulation. Zhu *et al* performed an experiment in which the two yeast transcriptional factors, Fkh1 and Fkh2[3], were knocked out and a time-series of gene expression levels was measured in alpha mating factor synchronized cells [60]. It is of interest to try to use this data source, along with Spellman *et al*'s gene expression time-series data from wildtype yeast and the DNA-binding data from Simon *et al*, to try to discover genes regulated by Fkh2. That is, by combining information from these data sources, one could examine genes determined to be bound by Fkh2 in wildtype yeast to see if these genes' expression patterns differ in the wildtype versus the mutant (knockout) yeast. However, direct comparison of the knockout data from Zhu *et al* [60] and the wildtype data from Spellman *et al* [51] is problematic, because the time-series were sampled at different rates, and also evidently have different periods and phases.

To study differential expression of genes between the wildtype and knockout experiments, the Fkh1/2DS was used as the reference set and the alphaDS was aligned with it. The non-uniform weighting version of the alignment algorithm described in section 3.2.1 was used to align the data sets. All of the approximately 800 genes identified to be cell cycle regulated as described in section 4.1 were used in the alignment.

The fifty-six genes determined to be bound by Fkh2 in Simon *et al*'s DNA-binding assay were then ranked according to the alignment error score $e_i^2(\widehat{\boldsymbol{\theta}})$. Figure 4-3 shows a plot of the spline expression profiles of the top four genes with the worst alignment scores and the top four with the best scores. A poor alignment score indicates that a gene is behaving differently in the knockout than it is in the wildtype experiment.

The ranking produced by the algorithm is biologically meaningful, highlighting which genes are likely to be regulated by Fkh2, versus those that are merely bound by it[4]. For instance, all of the genes shown with the worst alignment scores were determined to be bound by both Fkh1 and Fkh2 in [50], whereas all of the best aligning genes were determined to be bound by Fkh2 only. This corresponds to biological knowledge indicating that both Fkh1 and Fkh2 are required for regulation of a number of genes. It is also interesting that among the genes shown with good alignment scores, three of the four are bound also by Swi6 and either Mbp1 or Swi4, factors that are likely to work independently of the Fkh1/2 proteins. Further, all the genes shown with poor alignment scores are bound by the factors Ndd1 and Mcm1 or Ace2 and/or Swi5. Mcm1/Ndd1 are known to work with the Fkh1/2 proteins

[3]Fkh1 is a transcription factor with strong similarity to Fkh2, and the two are apparently partially redundant. So, unless both are knocked out, many effects of interest are not observed.

[4]The biological interpretation that follows is based on a personal communication from Dr. Itamar Simon.

and are not sufficient to regulate expression without Fkh1/2. Ace2 and Swi5 apparently can bind and regulate independently of the Fkh1/2 proteins, but their expression is Fkh1/2 dependent.

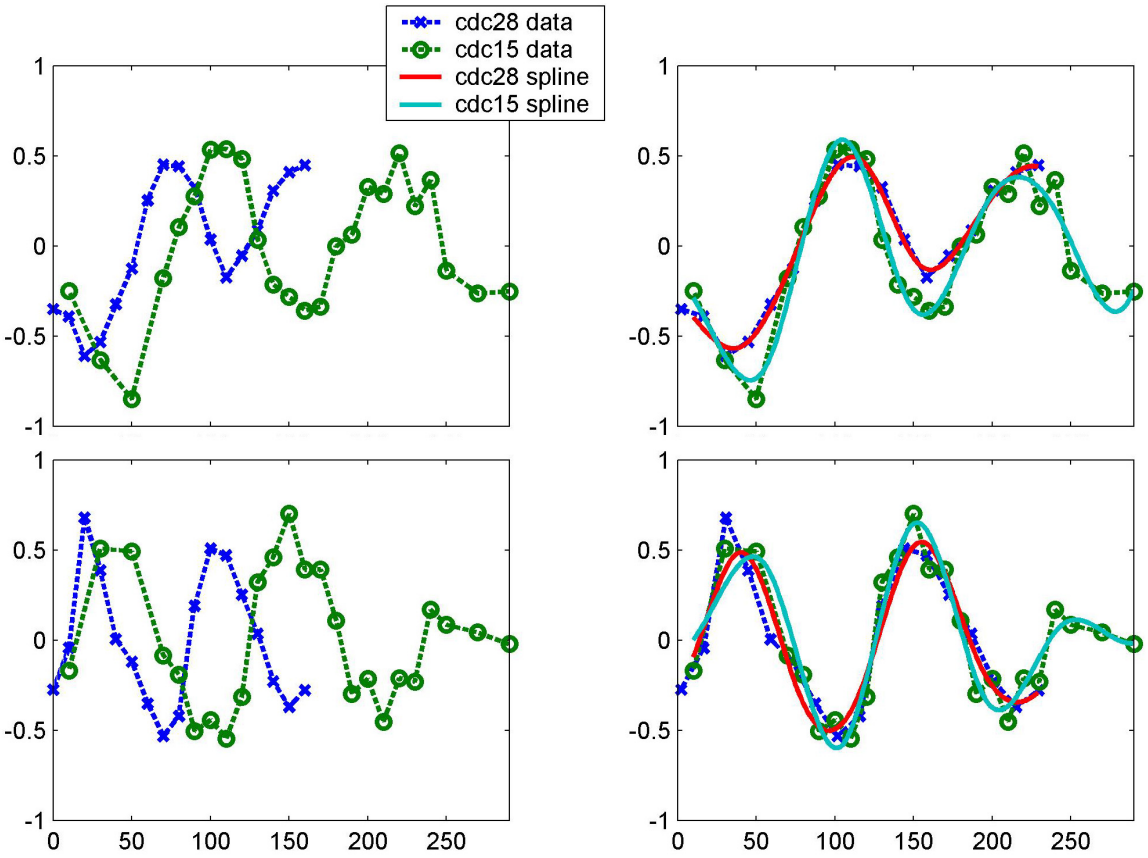

Figure 4-2: With the alignment algorithm it was estimated that the cdc28DS cell cycle runs at ≈ 1.4 times the speed of the cdc15DS cycle and starts ≈ 5.5 minutes before, a biologically reasonable result. Linear warping was used with the full set of approximately 800 cell cycle regulated genes. The left-hand side shows averages of unaligned expression values for genes in one of two phase classes. The right-hand side shows averages of the aligned predicted spline curves (and averages of the expression data values) for genes in one of two phase classes. The top row shows the $G1$ phase class (186 genes), and the bottom row the $S/G2$ phase class (283 genes).

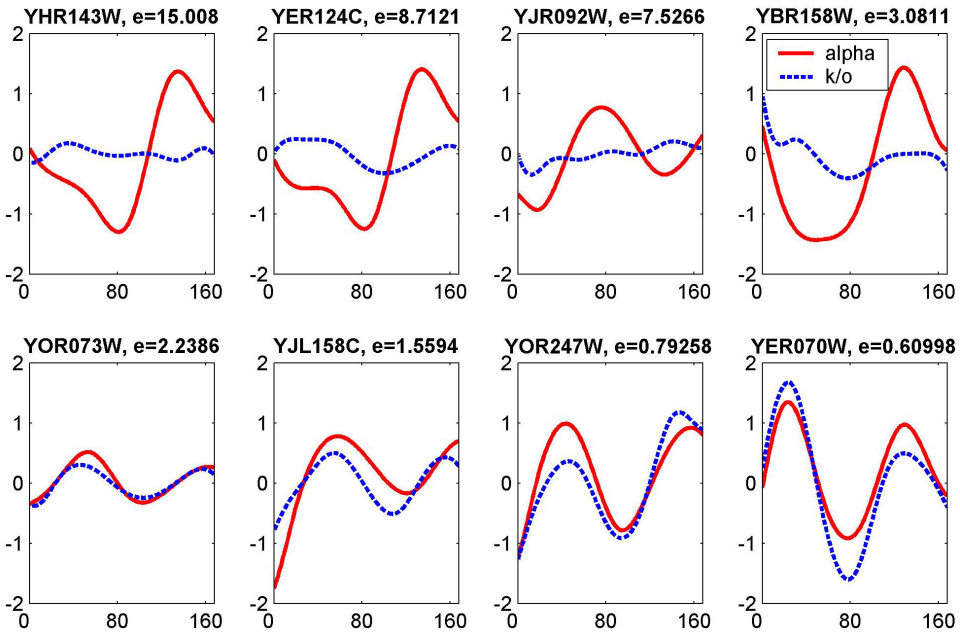

Figure 4-3: Genes that are regulated by Fkh2 can be identified by first aligning the Fkh1/2 knockout and the wildtype alpha mating factor synchronized data sets, and then ranking genes determined to be bound by Fkh2 (in a DNA-binding assay) according to their alignment error scores. Shown are the genes with the four worst (top row) and best (bottom row) alignment scores. A poor alignment score indicates that a gene is behaving differently in the knockout than it is in the wildtype experiment. See text for biological interpretation of these results.

Chapter 5

Conclusion

In this thesis, a probabilistic framework for continuous alignment of gene expression time-series data was presented. Gene expression time-series data were represented as a set of continuous spline curves (piecewise polynomials) using a linear mixed-effects model. This model constrains a gene's expression curve to be generally similar to those of co-expressed genes, while still allowing for gene specific variations. Estimation of the parameters of this model using the Expectation-Maximization (EM) algorithm was described. A continuous alignment technique was then introduced in which a search is performed for a function that transforms the time-scale of one gene expression process into that of another, while minimizing a measure of error between expression profiles in the two processes. Several improvements were then described for speeding up the alignment algorithm on large data sets.

Three applications were then presented of the continuous representation/alignment algorithm to DNA microarray expression data, which yielded biologically meaningful results. In the first application, it was shown that the linear mixed-effects spline model can effectively predict missing data values, achieving lower error for cases of one or two consecutive missing values than the best method previously reported in the literature. Second, it was demonstrated that the alignment algorithm can produce stable low-error alignments, allowing data from three different yeast cell cycle experiments (with period and phase differences) to be combined. Finally, it was shown that the alignment algorithm can be used to identify genes that behave differently in normal versus genetically modified cells.

5.1 Future Work

As more gene expression time-series data becomes available, I believe that the use of statistically principled continuous representation and alignment approaches such as the one presented in this thesis will become increasingly important, in order to understand dynamic genetic processes.

There are a number of interesting extensions that could be made to this work. For instance, it would seem natural to extend the statistical model so that class assignments can be learned automatically. In this thesis, classes for genes in the data sets analyzed were determined by using the method of Spellman *et al* [51], which uses prior biological knowledge for class assignments. In fact, it is possible to formulate a mixture model that can learn class assignments and obtain good results on the yeast cell cycle data set as demonstrated in the conference paper I co-authored [6]. However, for reasons of brevity and because the

alignment work was my primary contribution, I did not present these results in this thesis.

Another obvious extension would involve using non-linear warping functions, such as the monotone splines developed by Ramsay *et al* [44]. However, it should be noted that using more parameters increases the danger of overfitting, and currently available data sets have too few time-points to justify the use of non-linear warping.

Another extension would involve using different alignment error functions. The alignment error measure described in this thesis takes into account the difference in amplitude between the expression curves in the two experiments. In applications in which the amplitudes of the curves are quite different, another measure may be more appropriate, such as a continuous version of linear correlation that focuses more on similarity between the shapes of the curves.

Also, the random distribution of the spline coefficients could be considered in the alignment algorithm. Currently, the predicted spline curves are used. Instead, the curves could be treated as random variables and the alignment error measure could use an expectation of a function of these variables.

A method could also be developed for determining the significance of alignment errors so that differentially expressed genes could be automatically detected. Currently, the alignment errors are just used to rank genes and there is no specific criteria for determining differential expression. However, defining an appropriate distribution of the alignment errors remains an open question.

Finally, a different type of application could be explored in which genes exhibiting similar kinetic changes across experiments would be grouped together. That is, a clustering algorithm could be developed so that time-series of two different experiments measuring the same gene would be aligned and those genes that align best using similar warping parameters would be clustered together.

Appendix A

Overview of Molecular Genetics

This appendix will discuss the bare essentials of molecular genetics for those unfamiliar with the subject. For more information, consult a text such as Strachan and Read [52].

A.1 Essential Molecules For Life: DNA, RNA and Proteins

Much of cellular behavior arises from the interactions between three types of molecules: DNA (deoxyribonucleic acid), RNA (ribonucleic acid), and proteins. The so-called *central-dogma* of biology states that there is a one-way flow of biological information from DNA to RNA to proteins.

DNA may be thought of as the primary long-term information storage molecule in the cell. Often likened to the "blue-print" for the cell, DNA consists of a long backbone of alternating sugar and phosphate residues. Attached to each sugar residue is one of four nitrogen containing bases: adenine (A), cytosine (C), guanine (G) and thymine (T). It is the linear ordering or *sequence* of these bases in the DNA molecule that encodes information. Non-covalent bonds between hydrogen atoms in adenine/thymine and cytosine/guanine allow these bases to form stable pairs. The bases that form such pairs are said to be *complementary*. It is this base-pairing that allows DNA to exist as a double-stranded molecule in most normal physiological situations. The two strands encode the same information but use complementary bases.

While RNA also serves in an information carrying role in the cell, these molecules act in a more transitory manner than do DNA molecules. RNA is quite similar to DNA chemically, differing only in the type of sugar used (ribose instead of deoxyribose) and the use of the base uracil (U) instead of thymine. Unlike DNA, RNA molecules primarily exist in a single-stranded form, which is less stable and subject to cellular degradation. A primary function of cellular RNA is to act as the *messenger* molecules that carry information copied from DNA. This process of copying DNA to RNA is called *transcription*, and is carried out by pieces of cellular "machinery" called RNA polymerases. The single-stranded copies are called mRNA (messenger RNA) and serve to amplify the original DNA "signal." In keeping with the "blue-print" analogy, the RNA molecules are like working prints that have been photocopied from the original DNA plans.

Many mRNA copies are produced from the DNA, and the information encoded in these copies is then *translated* into proteins by elaborate pieces of cellular "machinery" called ribosomes. If mRNA copies are not made from a gene, then no protein will be produced and the gene is said to be silent or inactive. When the gene is transcribed into RNA copies,

the mRNA is said to be *expressed*.

Proteins are the primary "work-horses" of the cell, serving in a wide variety of roles including formation of physical structures, communication with other cells, and catalysis of metabolic reactions. In keeping with the "blue-print" analogy, proteins are the wood, walls, windows, doors, and even hammers and cranes used for constructing the building! Proteins consist of chains of amino acids, and the sequence of these amino acids and subsequent molecular modifications enable the protein to fold into a complex three-dimensional structure. While only four different nucleic acids are used to encode information at the DNA or mRNA level, proteins use twenty different amino acids. A triplet genetic code is used, in which three nucleotides, called a *codon*, specify a single amino acid. Since there are four types of nucleotides used in DNA, there are $4^3 = 64$ possible codons. This allows for a degenerate genetic code, in which a single amino acid can be specified by any of several codons.

A.2 Genes and Regulation

Because proteins play such a fundamental role in the life of a cell, biologists are extremely interested in identifying those regions of DNA called *genes* that ultimately lead to functional proteins. The complete set of genes in an organism is called its *genome*. A variety of computational and experimental techniques have been used to identify genes [52]. Surprisingly, in more advanced organisms, much DNA is never transcribed and even more mRNA is never translated. The explanations for this are complex and still not fully understood [47, 52]. For these and other reasons, discovering genes is by no means a straight-forward or error-free process. Further, even the definition of a gene is rather controversial and has been subject to revision in recent years [52]. Fortunately, there are organizations such as the National Center for Biotechnology Information that seek to compile databases of the latest consensus genetic data. Currently, information is publicly available for over four hundred organisms, including bacteria, yeast, worms, plants, rodents, and humans [37].

Genetic regulation — the control over when and where genes are active in the cell — is critical in all organisms. Since the ultimate products of genes are proteins that carry out an extremely diverse set of cellular tasks, lack of control of genetic activity can have disastrous consequences. For instance, in animals many genes code for proteins involved in cellular growth and division. If these genes are inappropriately regulated, cellular growth may go unchecked leading to malignant tumors. Since genetic regulation is so critical, its study comprises a large part of the field of molecular genetics.

The process of genetic regulation is very complex, and is not yet completely understood for even simple organisms such as bacterial viruses [41, 5]. On the most basic level, genetic regulation is carried out by transcriptional factors (specialized proteins) that act either to inhibit or activate transcription of genes. Many transcription factors bind to regions of DNA near the genes they regulate. In reality, the process of regulation is much more complicated, especially in more advanced organisms such as humans. Genetic regulation is known to involve combinatorial interactions among transcriptional factors, alterations of the three-dimensional structure of DNA, control of the rates of transcription and translation, and many other mechanisms [39, 34, 36, 18, 47, 41].

A.3 The "New Era" of Genomics

Classic molecular genetics tended to focus on the understanding of individual genes, which often involved years of painstaking experimental work for each gene [49]. It has only been in the last several years, with the advent of new automated technologies, that biologists have begun to take a more holistic approach. This approach is sometimes called *systems biology*, or more specifically *genomics*, when the goal is to understand organisms at the level of their genomes. This field is generally divided into two sub-disciplines: *structural* and *functional* genomics. Structural genomics is primarily concerned with elucidating the sequences of genes (order of nucleotides) and their positions relative to one another. The definition of functional genomics is a bit less clear, but an article on the topic by Philip Hieter and Mark Boguski [25] provides a useful viewpoint:

> "... functional genomics refers to the development and application of global (genome-wide or system-wide) experimental approaches to assess gene function by making use of the information and reagents provided by structural genomics. It is characterized by high throughput or large-scale experimental methodologies combined with statistical and computational analysis of the results ... Computational biology will perform a critical and expanding role in this area: whereas structural genomics has been characterized by data management, functional genomics will be characterized by mining the data sets for particularly valuable information."

Appendix B

DNA Microarrays

One of the most valuable tools in functional genomics is the DNA microarray, which allows researchers to measure the expression levels of thousands of mRNAs simultaneously. Although there are a variety of DNA microarray technologies in use, all operate on the basic principle that complementary strands of nucleic acids can "recognize" or hybridize with each other with high specificity through base-pairing.

DNA microarrays consist of a large number of different DNA molecules, called "probes," which are immobilized on a solid surface in an ordered matrix. The positions of the DNA molecules on the array are predetermined, and the different arrayed pieces of DNA usually correspond to portions of genes. Messenger RNA from a group of cells is then worked up to produce a sample of nucleic acid "targets," some of which will hybridize with the DNA probes on the microarray [49]. Various methods, usually involving either fluorescence or radioactivity, can be used to detect the degree of hybridization that occurs and thus quantify the expression levels of mRNAs present. The two most popular DNA microarray technologies differ principally in the probes or arrayed material used: either short sequences of DNA (oligonucleotides) [32, 3, 2] or complementary DNA (cDNA) [14, 9].

B.1 Photolithographic Oligonucleotide Arrays

One of the main producers of oligonucleotide arrays is the company Affymetrix, which uses a photolithographic process borrowed from semiconductor manufacturing [2]. An emerging alternative to the Affymetrix process uses ink-jet technology to deposit pre-synthesized oligonucleotides onto glass slides [24]. However, since this technology is not yet widely used it will not be discussed further in this thesis.

Affymetrix's photolithographic process begins with a quartz wafer that is coated with silane molecules, forming a uniform matrix. Nucleotides are then attached to this matrix via linker molecules. A photolithographic mask determines what area of the wafer will receive nucleotides. The mask has windows of 18 to 20 microns in size, and when ultraviolet light is shined through the windows, the exposed linkers become chemically "de-protected," allowing coupling to added nucleotides. The added nucleotides also have chemical groups at a particular position that are removed when exposed to ultraviolet light.

By using a sequence of different masks, and adding nucleotides in the appropriate order, a large number of oligonucleotides of chosen sequences may be synthesized directly on the quartz wafer. Current Affymetrix GeneChips© have over 500,000 spatially distinct "features" (oligonucleotides of different sequences) on a single 1.28 square centimeter array.

Each oligonucleotide is typically about 25 bases long.

Generally, 11 to 16 oligonucleotide probes are carefully chosen to represent a given gene transcript. Computational and empirical methods are used to choose the probes, in order to maximize sensitivity and specificity. For each probe selected, a *mismatch* probe is constructed that differs from the probe by one nucleotide near the center of the sequence. These mismatch probes serve as controls for non-specific hybridization. Thus, a total of 22 to 32 (match plus mismatch) probes are used for each gene. The state-of-the-art Affymetrix Human Genome U133 GeneChips© Set consists of two arrays that contain over one million unique oligonucleotide features corresponding to over 33,000 genes [2].

The sample or target is prepared by collecting mRNA from a population of cells and then using a viral enzyme, reverse transcriptase, to produce a DNA copy (cDNA) of the RNA[1]. The cDNA is then transcribed back into RNA *in vitro* using a source of ribonucleotides with attached biotin molecules. Biotin is a small molecule that binds very tightly to the protein streptavidin, which is used later in the microarray assay. The purpose of these steps is to amplify the amount of starting mRNA and to label the material. The labelled RNA is then hybridized to the microarray, and a fluorescent molecule attached to strepavadin is added. After several more staining and washing steps, the result is that areas on the chip will fluoresce proportionately to the concentration of hybridized RNA present [3].

A laser is used to excite the fluorescing molecules, and an optical scanner captures the emissions. Image processing algorithms are then used to correct for background noise and other optical effects [2]. The final output is a set of numbers that are related to the level of mRNA present for each gene.

B.2 Printed cDNA Arrays

In contrast to oligonucleotide arrays, cDNA arrays typically use much longer pieces of DNA (often a few hundred nucleotides in length [14, 9]). The DNA attached to these arrays is typically obtained from cDNA *libraries*. These libraries are usually constructed by making cDNA copies (with reverse transcriptase) of mRNA from particular cell types [14] (e.g., cells from different tissue sources). Once a set of cDNA probes has been selected, the typical procedure is to amplify the cDNA using the polymerase chain reaction (PCR) and then "print" it onto a glass microscope slide. Printing is done using a variety of technologies, although a typical setup uses a robotic arm that spots the DNA onto the slide using a pen-like apparatus [9]. Arrays have been constructed with this technique using over 30,000 different cDNAs [49]. The cDNA array technique is very flexible, since DNA from almost any source may be used. Specialized arrays have been constructed with cDNAs from human lymphocytes and other tissues, fruit flies, yeast, bacteria, and many other sources [4, 57, 51, 17].

As with the procedure for RNA preparation used for oligonucleotide arrays described above, the target material to be hybridized to a cDNA array consists of a sample of mRNA prepared from a population of cells. However, in the case of cDNA arrays, labelling is done during the reverse transcription to cDNA step. The *in vitro* transcription step is skipped, and the cDNA is hybridized directly to the microarray.

These is another important difference in target preparation that arises from the somewhat inexact printing process. Since inconsistent amounts of cDNA can be deposited at a given spot, direct comparisons across different arrays may be inaccurate. Thus, a method

[1]Yes, this does violate the central dogma, but biology is full of exceptions!

of competitive hybridization is used in which two separate samples labelled with different fluorescent dyes are simultaneously hybridized to the array.

The remaining laser excitation and scanning processes for cDNA arrays are also similar to those used for oligonucleotide arrays. One difference is that two lasers of different wavelengths are used, so that the intensity of fluorescence of the two samples can be measured separately. A ratio is then reported, giving the fold difference in expression between the two samples. Thus, with this technique the expression levels are always reported as ratios relative to one sample.

B.3 Comparison of Photolithographic Oligonucleotide and Printed cDNA Arrays

Both photolithographic oligonucleotide and printed cDNA microarrays are widely used, and have been applied to similar problems such as detecting genes involved in cancer or immune response [4, 20, 38, 8]. There are, however, a number of important differences between the two platforms. One technical concern is that using short segments of DNA can result in significant non-specific hybridization. However, the use of multiple probes covering different regions of a gene, mismatch probes, and recent work on using longer presynthesized oligonucleotides (50-100mers) [29], mitigate this problem to a large degree. A second technical concern is that oligonucleotide arrays require knowledge of a gene's DNA sequence. As an increasing amount of sequence information becomes available, this is likely to be less of an issue. But, there are certainly applications in which the flexibility of being able to use long pieces of DNA of unknown sequence could be useful.

However, probably the most important difference between the two platforms is not a technological one. Photolithographic oligonucleotide microarray production is much more costly and less flexible, in part due to the fact that manufacture is dominated by a single company, Affymetrix. In contrast, as mentioned above, printed cDNA microarrays have been manufactured in many different academic laboratories for diverse applications. In the future, as more competitors enter the arena, it is possible that mass-produced, low-cost, and highly accurate arrays will become commercially available.

B.4 Limitations of DNA Microarrays

Although DNA microarrays are an extremely important tool in functional genomics, they have a number of significant limitations. These limitations arise not only from the technologies, but also from inherent biological factors.

On a technical level, DNA microarray data is extremely noisy. Noise is introduced during the many steps of array manufacture, sample preparation, hybridization, and detection [23]. For instance, pipette error, temperature fluctuations, and reagent quality can introduce variation in mRNA amplification and the efficiency of fluorescent tagging. The variability of all these factors — from chip-to-chip, laboratory-to-laboratory, and even biology graduate student-to-graduate student — makes it challenging to compare results or to quantify precisely the detection limits of microarrays. For this reason, a large literature has developed presenting computational and experimental methods for reducing the effective noise level of microarrays (see for example the references [26, 54, 12, 29, 13]). A discussion of much of this literature is beyond the scope of this thesis, although some relevant references are mentioned in section 1.4.

From a biological perspective, an inherent limitation of using DNA microarrays to try to understand genetic function is that mRNA levels do not tell the whole story. Clearly, mRNA levels do provide some important information on the activity of a gene, since a gene must be transcribed into mRNA before it can be translated into a protein. However, the converse is not true: just because a gene is expressed does not mean that its ultimate protein product is active. Moreover, there may be a considerable time-lag between expression of a gene and the activity of its protein product. Additionally, the activity of proteins is further affected by chemical modifications and the binding of small molecules such as metabolites.

Thus, in order to understand more completely the cellular picture, we would need to measure many types of molecules engaged in numerous complex processes. Unfortunately, such an approach is limited by current technologies. Moreover, it is inherently more difficult to measure the levels of protein than it is to measure those of mRNAs, due to the complex structures of proteins. New technologies do provide some hope. For instance, researchers have recently used variants on DNA microarrays to measure the binding of proteins to regions of DNA [50] and levels of a limited number of proteins [61].

Appendix C

Derivations

C.1 Random Effects Predictor for the Linear Mixed-Effects Model

Suppose that we have two jointly Gaussian random vectors $\mathbf{x}$ and $\mathbf{y}$. It can be shown that the conditional distribution $\mathbf{x}|\mathbf{y}$ is a Gaussian [58], and it can be written in terms of the statistics of the two random vectors. Denote the mean of the conditional distribution by $\boldsymbol{m}_{\mathbf{x}|\mathbf{y}}(\boldsymbol{y})$ and the variance of this distribution by $\boldsymbol{\Lambda}_{\mathbf{x}|\mathbf{y}}$. We can then write the statistics of the conditional distribution as:

$$\boldsymbol{m}_{\mathbf{x}|\mathbf{y}}(\boldsymbol{y}) = \boldsymbol{m}_{\mathbf{x}} + \boldsymbol{\Lambda}_{\mathbf{xy}}\boldsymbol{\Lambda}_{\mathbf{y}}^{-1}(\boldsymbol{y} - \boldsymbol{m}_{\mathbf{y}}) \tag{C.1}$$

$$\boldsymbol{\Lambda}_{\mathbf{x}|\mathbf{y}} = \boldsymbol{\Lambda}_{\mathbf{x}} - \boldsymbol{\Lambda}_{\mathbf{xy}}\boldsymbol{\Lambda}_{\mathbf{y}}^{-\mathbf{1}}\boldsymbol{\Lambda}_{\mathbf{xy}}^{\mathbf{T}} \tag{C.2}$$

Now, for the linear mixed-effects model described in section 2.1, we have that $\mathbf{y}_i \sim N(\boldsymbol{A}_i\boldsymbol{\mu}, \boldsymbol{B}_i\boldsymbol{\Gamma}\boldsymbol{B}_i^T + \sigma^2\boldsymbol{I})$ and $\boldsymbol{\gamma}_i \sim N(\mathbf{0}, \boldsymbol{\Gamma})$. We can then derive the cross-covariance matrix $\boldsymbol{\Lambda}_{\boldsymbol{\gamma}_i\mathbf{y}_i}$ as follows:

$$\begin{aligned}\boldsymbol{\Lambda}_{\boldsymbol{\gamma}_i\mathbf{y}_i} = E[(\boldsymbol{\gamma}_i - \boldsymbol{m}_{\boldsymbol{\gamma}_i})(\mathbf{y}_i - \boldsymbol{m}_{\mathbf{y}_i})^T] = E[\boldsymbol{\gamma}_i(\mathbf{y}_i - \boldsymbol{A}_i\boldsymbol{\mu})^T] = \\ E[\boldsymbol{\gamma}_i\boldsymbol{\gamma}_i^T\boldsymbol{B}_i^T + \boldsymbol{\gamma}_i\boldsymbol{\epsilon}_i^T] = \\ E[\boldsymbol{\gamma}_i\boldsymbol{\gamma}_i^T\boldsymbol{B}_i^T] = \boldsymbol{\Gamma}\boldsymbol{B}_i^T\end{aligned} \tag{C.3}$$

In the above, we used the fact that $E[\boldsymbol{\gamma}_i] = \mathbf{0}$ and that $\boldsymbol{\gamma}_i$ is uncorrelated with $\boldsymbol{\epsilon}_i$. Then, using equations C.1 and C.3 we can write the predictor as:

$$\widehat{\boldsymbol{\gamma}}_i(\boldsymbol{y}_i) = \boldsymbol{m}_{\boldsymbol{\gamma}_i}(\boldsymbol{y}_i) = \boldsymbol{\Gamma}\boldsymbol{B}_i^T(\boldsymbol{B}_i\boldsymbol{\Gamma}\boldsymbol{B}_i^T + \sigma^2\boldsymbol{I})^{-1}(\boldsymbol{y}_i - \boldsymbol{A}_i\boldsymbol{\mu}) \tag{C.4}$$

Finally, it can be shown that the conditional covariance matrix is:

$$\text{var}(\boldsymbol{\gamma}_i|\mathbf{y}_i) = \boldsymbol{\Lambda}_{\boldsymbol{\gamma}_i|\mathbf{y}_i} = (\boldsymbol{\Gamma}^{-1} + \boldsymbol{B}_i^T\boldsymbol{B}_i/\sigma^2)^{-1} \tag{C.5}$$

This follows from equations C.2 and C.3:

$$\begin{aligned}\boldsymbol{\Lambda}_{\boldsymbol{\gamma}_i|\mathbf{y}_i} = \boldsymbol{\Lambda}_{\boldsymbol{\gamma}_i} - \boldsymbol{\Lambda}_{\boldsymbol{\gamma}_i\mathbf{y}_i}\boldsymbol{\Lambda}_{\mathbf{y}_i}^{-1}\boldsymbol{\Lambda}_{\boldsymbol{\gamma}_i\mathbf{y}_i}^T = \\ \boldsymbol{\Gamma} - \boldsymbol{\Gamma}\boldsymbol{B}_i^T(\boldsymbol{B}_i\boldsymbol{\Gamma}\boldsymbol{B}_i^T + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{B}_i\boldsymbol{\Gamma}^T = \\ \{\boldsymbol{\Gamma}(\boldsymbol{B}_i\boldsymbol{\Gamma})^{-1}(\boldsymbol{B}_i\boldsymbol{\Gamma}\boldsymbol{B}_i^T + \sigma^2\boldsymbol{I}) - \boldsymbol{\Gamma}\boldsymbol{B}_i^T\}\{(\boldsymbol{B}_i\boldsymbol{\Gamma}\boldsymbol{B}_i^T + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{B}_i\boldsymbol{\Gamma}\} =\end{aligned}$$

$$(\sigma^2 \boldsymbol{B}_i^{-1})(\boldsymbol{B}_i \boldsymbol{\Gamma} \boldsymbol{B}_i^T + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{B}_i \boldsymbol{\Gamma} =$$
$$(\sigma^2 \boldsymbol{B}_i^{-1})[(\boldsymbol{\Gamma}^{-1} \boldsymbol{B}_i^{-1})(\boldsymbol{B}_i \boldsymbol{\Gamma} \boldsymbol{B}_i^T + \sigma^2 \boldsymbol{I})]^{-1} =$$
$$\sigma^2 [(\boldsymbol{B}_i^T + \sigma^2 \boldsymbol{\Gamma}^{-1} \boldsymbol{B}_i^{-1}) \boldsymbol{B}_i]^{-1} =$$
$$(\boldsymbol{\Gamma}^{-1} + \boldsymbol{B}_i^T \boldsymbol{B}_i / \sigma^2)^{-1}$$

C.2 Estimating Parameters of the Linear Mixed-Effects Model Using the EM Algorithm

This section follows the method outlined in [6, 27, 28], but gives a more detailed derivation. Note that using the penalized least squares method yields an identical solution (see [28] for details).

The EM algorithm is initialized with a parameter vector $\widehat{\boldsymbol{\Psi}}^{(0)} = (\boldsymbol{\mu}^{(0)}, \boldsymbol{\Gamma}^{(0)}, \sigma^{(0)})^T$. Random values could be used, or other possibilities such as the mean of the observed data for $\boldsymbol{\mu}^{(0)}$, the identity matrix for $\boldsymbol{\Gamma}^{(0)}$ and unity for $\boldsymbol{\sigma}^{(0)}$.

For the linear mixed-effects model described in section 2.1, the complete data likelihood can be factored in a convenient way:

$$L_c(\boldsymbol{\Psi}) = p(\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_n, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_n; \boldsymbol{\Psi}) =$$
$$p(\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_n; \boldsymbol{\Psi}) p(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n | \boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_n; \boldsymbol{\Psi})$$

As was noted in section 2.1, $\mathbf{y}_i | \boldsymbol{\gamma}_i \sim N(\boldsymbol{A}_i \boldsymbol{\mu} + \boldsymbol{B}_i \boldsymbol{\gamma}_i, \sigma^2 \boldsymbol{I})$ and $\boldsymbol{\gamma}_i \sim (\mathbf{0}, \boldsymbol{\Gamma})$. Hence, the complete data likelihood is:

$$L_c(\boldsymbol{\Psi}) = \prod_{i=1}^{n} \frac{1}{(2\pi)^{m/2} \sigma^m} \times$$
$$\exp[-\frac{1}{2\sigma^2} (\boldsymbol{y_i} - \boldsymbol{A}_i \boldsymbol{\mu} - \boldsymbol{B}_i \boldsymbol{\gamma}_i)^T (\boldsymbol{y_i} - \boldsymbol{A}_i \boldsymbol{\mu} - \boldsymbol{B}_i \boldsymbol{\gamma}_i)] \times$$
$$\frac{1}{(2\pi)^{m/2} [\det(\boldsymbol{\Gamma})]^{1/2}} \exp[-\frac{1}{2} \boldsymbol{\gamma}_i^T \boldsymbol{\Gamma}^{-1} \boldsymbol{\gamma}_i]$$

Taking the logarithm of this equation yields:

$$\log L_c(\boldsymbol{\Psi}) = -nm \log \sigma - \frac{1}{2\sigma^2} \sum_{i=1}^{n} \|\boldsymbol{y}_i - \boldsymbol{A}_i \boldsymbol{\mu} - \boldsymbol{B}_i \boldsymbol{\gamma}_i\|^2 \tag{C.6}$$
$$-\frac{n}{2} \log \det(\boldsymbol{\Gamma}) - \frac{1}{2} \sum_{i=1}^{n} \boldsymbol{\gamma}_i^T \boldsymbol{\Gamma}^{-1} \boldsymbol{\gamma}_i + C$$

Here, C is a constant that does not depend on any of the parameters or random variables. For the E-step, the expectation of the $\log L_c(\boldsymbol{\Psi})$ function is taken with respect to the $\boldsymbol{\gamma}_i$ random variables conditioned on the observed data. Inspection of equation C.6 reveals that there are terms linear in $\boldsymbol{\gamma}_i$, as well as the quadratic terms: $\boldsymbol{\gamma}_i^T \boldsymbol{B}_i^T \boldsymbol{B}_i \boldsymbol{\gamma}_i / \sigma^2$ and $\boldsymbol{\gamma}_i^T \boldsymbol{\Gamma}^{-1} \boldsymbol{\gamma}_i$. For the linear terms, we can simply replace $\boldsymbol{\gamma}_i$ with the conditional expectation $\widehat{\boldsymbol{\gamma}}_i$ as derived in section C.1. For the quadratic terms, we will make use of a general formula. Suppose $\boldsymbol{F}$

is a square q by q matrix. Then we can write:

$$E[\gamma_i^T \boldsymbol{F} \gamma_i | \mathbf{y}_i] = \sum_{j=1}^{q}\sum_{l=1}^{q}[F]_{jl} E\{[\gamma_i]_j [\gamma_i]_l | \mathbf{y}_i\} =$$
$$\widehat{\gamma}_i^T \boldsymbol{F} \widehat{\gamma}_i + \sum_{j=1}^{q}\sum_{l=1}^{q}[F]_{jl}[\Lambda_{\gamma_i|\mathbf{y_i}}]_{jl} =$$
$$\widehat{\gamma}_i^T \boldsymbol{F} \widehat{\gamma}_i + \text{trace}(\boldsymbol{\Lambda}_{\gamma_i|\mathbf{y}_i} \boldsymbol{F})$$

Using the fact that $E_{\boldsymbol{\Psi}^{(k)}}[\gamma_i \gamma_i^T | \mathbf{y}_i] = \gamma_i^{(k)} (\gamma_i^{(k)})^T + \boldsymbol{\Lambda}^k_{\gamma_i|\mathbf{y}_i} = \gamma_i^{(k)} (\gamma_i^{(k)})^T + ((\boldsymbol{\Gamma}^k)^{-1} + \boldsymbol{B}_i^T \boldsymbol{B}_i / (\sigma^{(k)})^2)^{-1}$ from section C.1 and the above formula, we can calculate the expectations for the quadratic terms:

$$E_{\boldsymbol{\Psi}^{(k)}}[\gamma_i^T \boldsymbol{B}_i^T \boldsymbol{B}_i \gamma_i / \sigma^2 | \mathbf{y}_i] = \tag{C.7}$$
$$(\widehat{\gamma}_i^{(k)})^T \boldsymbol{B}_i^T \boldsymbol{B}_i \widehat{\gamma}_i^{(k)} / \sigma^2 + \text{trace}(\boldsymbol{\Lambda}^{(k)}_{\gamma_i|\mathbf{y}_i} \boldsymbol{B}_i^T \boldsymbol{B}_i)/\sigma^2$$

$$E_{\boldsymbol{\Psi}^{(k)}}[\gamma_i^T \boldsymbol{\Gamma}^{-1} \gamma_i | \mathbf{y}_i] = (\widehat{\gamma}_i^{(k)})^T \boldsymbol{\Gamma}^{-1} \widehat{\gamma}_i^{(k)} + \text{trace}(\boldsymbol{\Lambda}^{(k)}_{\gamma_i|\mathbf{y}_i} \boldsymbol{\Gamma}^{-1}) \tag{C.8}$$

These results allow us to calculate the conditional expectation of the complete log likelihood in the k^{th} E-step as:

$$E_{\widehat{\boldsymbol{\Psi}}^{(k)}}[\log L_c(\boldsymbol{\Psi}) | \mathbf{y}] = -nm \log \sigma - \frac{1}{2\sigma^2} \sum_{i=1}^{n} \| \boldsymbol{y}_i - \boldsymbol{A}_i \boldsymbol{\mu} - \boldsymbol{B}_i \widehat{\gamma}_i^{(k)} \|^2$$
$$-\frac{n}{2} \log \det(\boldsymbol{\Gamma}) - \frac{1}{2} \sum_{i=1}^{n} (\widehat{\gamma}_i^{(k)})^T \boldsymbol{\Gamma}^{-1} \widehat{\gamma}_i^{(k)}$$
$$-\frac{1}{2\sigma^2} \sum_{i=1}^{n} \text{trace}(\boldsymbol{\Lambda}^{(k)}_{\gamma_i|\mathbf{y}_i} \boldsymbol{B}_i^T \boldsymbol{B}_i)$$
$$-\frac{1}{2} \sum_{i=1}^{n} \text{trace}(\boldsymbol{\Lambda}^{(k)}_{\gamma_i|\mathbf{y}_i} \boldsymbol{\Gamma}^{-1}) + C$$

The parameters can now be maximized in the $(k+1)^{th}$ M-step by taking partial derivatives and equating them to zero. The solutions are as follows:

$$\boldsymbol{\mu}^{(k+1)} = (\sum_{i=1}^{n} \boldsymbol{A}_i^T \boldsymbol{A}_i)^{-1} [\sum_{i=1}^{n} \boldsymbol{A}_i^T (\boldsymbol{d}_i - \boldsymbol{B}_i \widehat{\gamma}_i^k)] \tag{C.9}$$

$$(\sigma^{(k+1)})^2 = \frac{1}{nm} \sum_{i=1}^{n} \| \boldsymbol{d}_i - \boldsymbol{A}_i \boldsymbol{\mu}^{(k+1)} - \boldsymbol{B}_i \widehat{\gamma}_i^{(k)} \|^2 + \tag{C.10}$$
$$\frac{1}{nm} \sum_{i=1}^{n} \text{trace}(\boldsymbol{\Lambda}^{(k)}_{\gamma_i|\mathbf{y}_i} \boldsymbol{B}_i^T \boldsymbol{B}_i)$$

$$\boldsymbol{\Gamma}^{(k+1)} = \frac{1}{n} \sum_{i=1}^{n} [\widehat{\gamma}_i^{(k)} (\widehat{\gamma}_i^{(k)})^T + \boldsymbol{\Lambda}^{(k)}_{\boldsymbol{\gamma_i}|\mathbf{y}_i}] \tag{C.11}$$

Notice that maximizing the parameters amounts almost just to "plugging" the predicted

values $\widehat{\gamma}_i$ into the typical maximum likelihood (ML) estimation formulas for a multivariate Gaussian. However, the formulas for σ and $\mathbf{\Gamma}$ have "correction" terms added. Intuitively, one can think of these terms as correcting for the additional uncertainty in the predictions $\widehat{\gamma}_i$.

C.2.1 Modifications Necessary for the Gene Expression Time-Series Linear Mixed-Effects Spline Model

The E- and M-steps for estimating parameters of the model presented in section 2.4 are almost identical to those discussed above. The only real difference is that we change notation, letting $\boldsymbol{S}_i = \boldsymbol{A}_i = \boldsymbol{B}_i$ and we now must estimate parameters $\boldsymbol{\mu}_j$ and $\mathbf{\Gamma}_j$ for each class C_j. The complete data likelihood for the model presented in section 2.4 factors as follows:

$$\begin{aligned} L_c(\mathbf{\Psi}) = p(\gamma_1, \ldots, \gamma_{n_G}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{n_G}; \mathbf{\Psi}) = \\ p(\gamma_1, \ldots, \gamma_{n_G}; \mathbf{\Psi}) p(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{n_G} | \gamma_1, \ldots, \gamma_{n_G}; \mathbf{\Psi}) = \\ \prod_{i=1}^{n_G} p(\gamma_i; \mathbf{\Psi}) p(\boldsymbol{y}_i | \gamma_i; \mathbf{\Psi}) = \\ \prod_{j=1}^{n_C} \prod_{i \in C_j} p(\gamma_i; \boldsymbol{\mu}_j, \mathbf{\Gamma}_j, \sigma) p(\boldsymbol{y}_i | \gamma_i; \boldsymbol{\mu}_j, \mathbf{\Gamma}_j, \sigma) \end{aligned}$$

Plugging in the distributions of the random variables, we get:

$$\begin{aligned} L_c(\mathbf{\Psi}) = \prod_{j=1}^{n_C} \prod_{i \in C_j} \frac{1}{(2\pi)^{n_{T_i}/2} \sigma^{n_{T_i}}} \times \\ \exp[-\frac{1}{2\sigma^2} (\boldsymbol{y_i} - \boldsymbol{S}_i \boldsymbol{\mu}_j - \boldsymbol{S}_i \gamma_i)^T (\boldsymbol{y_i} - \boldsymbol{S}_i \boldsymbol{\mu}_j - \boldsymbol{S}_i \gamma_i)] \times \\ \frac{1}{(2\pi)^{n_{T_i}/2} [\det(\mathbf{\Gamma}_j)]^{1/2}} \exp[-\frac{1}{2} \gamma_i^T \mathbf{\Gamma}_j^{-1} \gamma_i] \end{aligned}$$

Taking the logarithm of this equation yields:

$$\begin{aligned} \log L_c(\mathbf{\Psi}) = -\sum_{j=1}^{n_C} \sum_{i \in C_j} n_{T_i} \log \sigma - \frac{1}{2\sigma^2} \sum_{j=1}^{n_C} \sum_{i \in C_j} \| \boldsymbol{y}_i - \boldsymbol{S}_i \boldsymbol{\mu}_j - \boldsymbol{S}_i \gamma_i \|^2 \\ -\frac{1}{2} \sum_{j=1}^{n_C} n_{C_j} \log \det(\mathbf{\Gamma}_j) - \frac{1}{2} \sum_{j=1}^{n_C} \sum_{i \in C_j} \gamma_i^T \mathbf{\Gamma}_j^{-1} \gamma_i + C \end{aligned} \tag{C.12}$$

As before, the parameters are maximized in the $(k+1)^{th}$ M-step by taking partial derivatives and equating them to zero. The solutions are as follows:

$$\boldsymbol{\mu}_j^{(k+1)} = (\sum_{i \in C_j} \boldsymbol{S}_i^T \boldsymbol{S}_i)^{-1} [\sum_{i \in C_j} \boldsymbol{S}_i^T (\boldsymbol{d}_i - \boldsymbol{S}_i \widehat{\gamma}_i^k)] \tag{C.13}$$

$$(\sigma^{(k+1)})^2 = \frac{1}{\sum_{j=1}^{n_C} \sum_{i \in C_j} n_{T_i}} \sum_{j=1}^{n_C} \sum_{i \in C_j} \{ \| \boldsymbol{d}_i - \boldsymbol{S}_i \boldsymbol{\mu}_j^{(k+1)} - \boldsymbol{S}_i \widehat{\gamma}_i^{(k)} \|^2 + \tag{C.14}$$

$$\text{trace}((\boldsymbol{\Lambda}^{(k)}_{\boldsymbol{\gamma}_i|\mathbf{y}_i})_j \boldsymbol{S}_i^T \boldsymbol{S}_i)\}$$

$$\boldsymbol{\Gamma}_j^{(k+1)} = \frac{1}{n_{C_j}} \sum_{i \in C_j} [\widehat{\boldsymbol{\gamma}}_i^{(k)} (\widehat{\boldsymbol{\gamma}}_i^{(k)})^T + (\boldsymbol{\Lambda}^{(k)}_{\boldsymbol{\gamma}_i|\mathbf{y}_i})_j] \tag{C.15}$$

C.3 Expanded Form of the Spline Predictor

The spline predictor in section 2.4 is given as:

$$\widehat{y}_i(t) = [\boldsymbol{s}(t)]^T (\boldsymbol{\mu}_j + \widehat{\boldsymbol{\gamma}}_i) \tag{C.16}$$

Using the fact that $\widehat{\boldsymbol{\gamma}}_i(\boldsymbol{d}_i) = \boldsymbol{\Gamma}_j \boldsymbol{S}_i^T (\boldsymbol{S}_i \boldsymbol{\Gamma}_j \boldsymbol{S}_i^T + \sigma^2 \boldsymbol{I})^{-1} (\boldsymbol{d}_i - \boldsymbol{S}_i \boldsymbol{\mu}_j)$ we can expand equation C.16 to get:

$$\begin{aligned}
\widehat{y}_i(t) = [\boldsymbol{s}(t)]^T [\boldsymbol{\mu}_j + \boldsymbol{\Gamma}_j \boldsymbol{S}_i^T (\boldsymbol{S}_i \boldsymbol{\Gamma}_j \boldsymbol{S}_i^T + \sigma^2 \boldsymbol{I})^{-1} (\boldsymbol{d}_i - \boldsymbol{S}_i \boldsymbol{\mu}_j)] = \\
[\boldsymbol{s}(t)]^T \{[\boldsymbol{I} - \boldsymbol{\Gamma}_j \boldsymbol{S}_i^T (\boldsymbol{S}_i \boldsymbol{\Gamma}_j + \boldsymbol{S}_i^T)^{-1} \boldsymbol{S}_i] \boldsymbol{\mu}_j + \boldsymbol{\Gamma}_j \boldsymbol{S}_i^T (\boldsymbol{S}_i \boldsymbol{\Gamma}_j \boldsymbol{S}_i^T + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{d}_i\} = \\
[\boldsymbol{s}(t)]^T \{[\boldsymbol{I} - (\boldsymbol{S}_i^T \boldsymbol{S}_i + \sigma^2 \boldsymbol{\Gamma}_j^{-1})^{-1} \boldsymbol{S}_i^T \boldsymbol{S}_i] \boldsymbol{\mu}_j + (\boldsymbol{S}_i^T \boldsymbol{S}_i + \sigma^2 \boldsymbol{\Gamma}_j^{-1})^{-1} \boldsymbol{S}_i^T \boldsymbol{d}_i\}
\end{aligned}$$

Bibliography

[1] J. Aach and G. M. Church. Aligning gene expression time series with time warping algorithms. *Bioinformatics*, 17:495–508, 2001.

[2] Affymetrix, Inc., website. *www.affymetrix.com*.

[3] Affymetrix, Inc., Santa Clara, CA 95051. *GeneChip Expression Analysis Technical Manual*, 2001.

[4] A.A. Alizadeh and et al. Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Science*, 403:503–510, 2000.

[5] A. Arkin, J. Ross, and H.H. McAdams. Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected escherichia coli cells. *Genetics*, 149(4):1633–48, 1998.

[6] Z. Bar-Joseph, G. Gerber, D. Gifford, T. Jaakkola, and I. Simon. A new approach to analyzing gene expression time-series data. In *RECOMB*, 2002.

[7] R. Bartels, J. Beatty, and B. Barsky. *Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufman, 1987.

[8] J.C. Boldrick and et al. Stereotyped and specific gene expression programs in human innate immune responses to bacteria. *PNAS*, 99:927–77, 2002.

[9] C.G. Cheung and et al. Making and reading microarrays. *Nature Genetics Supplement*, 21:15–19, 1999.

[10] L. Deng, M. Aksmanovic, D. X. Sun, and C. F. J. X. Wu. Recognition using hidden markov models with polynomial regression functions as nonstationary states. *IEEE Transactions on Speech and Audio Processing*, 2:507–520, 1994.

[11] P. D'haeseleer, X. Wen, S. Fuhrman, and R. Somogyi. Linear modeling of mrna expression levels during cns development and injury. In *PSB99*, 1999.

[12] R. Dror, J. Murnick, and et al. A bayesian approach to transcript estimation from gene array data: the beam technique. In *RECOMB*, 2002.

[13] A.M. Dudley, J. Aach, M.A. Steffen, and G.M. Church. Measuring absolute expression with microarrays with calibrated reference sample and an extended signal intensity range. *PNAS*, 99:7554–7559, 2002.

[14] D.J. Duggan and et al. Expression profiling using cdna microarrays. *Nature Genetics Supplement*, 21:10–14, 1999.

[15] M.B. Eisen, P.T. Spellman, P.O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *PNAS*, 95:14863–14868, 1998.

[16] R. Eubank. *Nonparametric regression and spline smoothing*. Marcel Dekker, 1999.

[17] M.A. Fisher, B.B. Plikaytis, and T.M. Shinnick. Microarray analysis of the mycobacterium tuberculosis transcriptional response to the acidic conditions found in phagosomes. *J. Bacteriology*, 184:4025–32, 2002.

[18] M.C. Fonseca. The contribution of nuclear compartmentalization to gene regulation. *Cell*, 108:513–521, 2002.

[19] N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using bayesian network to analyze expression data. In *RECOMB*, 2000.

[20] T. Golub and et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

[21] A. J. Hartemink, D. K. Gifford, T. S. Jaakkola, and R.A. Young. Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. In *PSB01*, 2001.

[22] A. J. Hartemink, D. K. Gifford, T. S. Jaakkola, and R.A. Young. Combining location and expression data for principled discovery of genetic regulatory network models. In *PSB02*, 2002.

[23] A.J. Hartemink. *Principled Computational Methods for the Validation and Discovery of Genetic Regulatory Networks*. PhD thesis, Massachusetts Institute of Technology, 2001.

[24] M.J. Heller. Dna microarray technology: Devices, systems, and applications. *Annu. Rev. Biomed. Eng.*, 4:129–53, 2002.

[25] P. Hieter and M. Boguski. Functional genomics: It's all how you read it. *Science*, 278:601–602, 1997.

[26] T. Ideker, V. Thorsson, and et al. Testing for differentially-expressed genes by maximum-likelihood analysis of microarray data. *Journal of Computational Biology*, 7:805–817, 2000.

[27] G. James and T. Hastie. Functional linear discriminant analysis for irregularly sampled curves. *Journal of the Royal Statistical Society, Series B*, to appear, 2001.

[28] G. James, T. Hastie, and C. Sugar. Principal component models for sparse functional data. *Biometrika*, 87:587–602, 2000.

[29] M.D. Kane and et al. Assessment of the sensitivity and specificity of oligonucleotide (50mer) microarrays. *Nucleic Acids Research*, 28:4552–4557, 2000.

[30] N. Laird and J. Ware. Random-effects models for longitudinal data. *Biometrics*, 38:963–974, 1982.

[31] C. Lawrence, J.L. Zhou, and A.L. Tits. *User's Guide for CFSQP Version 2.5*. Institute for Systems Research, University of Maryland, College Park, MD 20742, 1997.

[32] R.J. Lipshutz and et al. High density synthetic oligonucleotide arrays. *Nature Genetics Supplement*, 21:20–24, 1999.

[33] The MathWorks, Inc., Natick, MA, USA. *Matlab Optimization Toolbox 2.1 User's Guide*, 2000.

[34] N.J. McKenna and B.W. O'Malley. Combinatorial control of gene expression by nuclear receptors and coregulators. *Cell*, 108:465–474, 2002.

[35] G.J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. John Wiley & Sons, 1997.

[36] G.J. Narlikar, H.Y. Fan, and R.E. Kingston. Cooperation between complexes that regulate chromatin structure and transcription. *Cell*, 108:475–487, 2002.

[37] National Center for Biotechnology Information, website. *www.ncbi.nlm.nih.gov*.

[38] G. Nau, J. Richmond, and et al. Human macrophage activation programs induced by bacterial pathogens. *PNAS*, 99(3):1503–8, 2002.

[39] G. Orphanides and D. Reinberg. A unified theory of gene expression. *Cell*, 108:439–451, 2002.

[40] E.R. Panier and A.L. Tits. On combining feasibility, descent and superlinear convergence in inequality constrained optimization. *Math. Programming*, 59:261–276, 1993.

[41] M. Ptashne and E. Ganna. *Genes and Signals*. Cold Spring Habor Laboratory Press, 2002.

[42] J. Qian, M. Dolled-Filhart, J. Lin, H. Yu, and M. Gerstein. Beyond synexpression relationships: Local clustering of time-shifted and inverted gene expression profiles indentifies new, biologically relevant interactions. *Journal of Molecular Biology*, 314:1053–1066, 2001.

[43] M. Ramoni, P. Sebastiani, and I. Kohane. Cluster analysis of gene expression dynamics. *PNAS*, 99:9121–9126, 2002.

[44] J. Ramsay. Estimating smooth monotone functions. *Journal of the Royal Statistical Society, Series B*, 60:365–375, 1998.

[45] J. Ramsay and X. Li. Curve registration. *Journal of the Royal Statistical Society, Series B*, 60:351–363, 1998.

[46] B. Reis, A. Butte, and I. Kohane. Extracting knowledge from dynamics in gene expression. *Journal of Biomedical Informatics*, pages 1–13, 2001.

[47] S.C.R. Richards, E.J. amd Elgin. Epigenetic codes for heterochromatin formation and silencing: Rounding up the usual suspects. *Cell*, 108:489–500, 2002.

[48] D. Rogers and J. Adams. *Mathematical Elements for Computer Graphics*. McGraw-Hill, 1990.

[49] A. Schulze and J. Downward. Navigating gene expression using microarrays - a technology review. *Nature Cell Biology*, 3:190–195, 2001.

[50] I. Simon, J. Barnett, and et al. Serial regulation of transcriptional regulators in the yeast cell cycle. *Cell*, 106:697–708, 2001.

[51] T. S. Spellman, G. Sherlock, and et al. Comprehensive identification of cell cycle-regulated genes of the yeast *saccharomyces cerevisia* by microarray hybridization. *Mol. Biol. of the Cell*, 9:3273–3297, 1998.

[52] T. Strachan and A.P. Read. *Human Molecular Genetics*. John Wiley & Sons, second edition, 1999.

[53] O. Troyanskaya, M. Cantor, and et al. Missing value estimation methods for dna microarrays. *Bioinformatics*, 17:520–525, 2001.

[54] G. Tusher, R. Tibshirani, and G. Chu. Significance analysis of microarrays applied to the ionizing radiation response. *PNAS*, 98(9):5162–5121, 2001.

[55] P. Viola. *Alignment by Maximization of Mutual Information*. PhD thesis, MIT AI Lab, 1995.

[56] A. Watt and M. Watt. *Advanced Animation and Rendering Techniques*. Addison-Wesley, 1992.

[57] K. White, S. Rifkin, and et al. Microarray analysis of drosophila development during metamorphosis. *Science*, 286:2179–2184, 1999.

[58] A.S. Willsky, G.W. Wornell, and J.H. Shapiro. *6.432: Stochastic Processes, Detection and Estimation*. MIT Course Notes, 2002.

[59] L. P. Zhao, R. Prentice, and L. Breeden. Statistical modeling of large microarray data sets to identify stimulus-response profiles. *PNAS*, 98:5631–5636, 2001.

[60] G. Zhu, Spellman T. S., and et al. Two yeast forkhead genes regulate cell cycle and pseudohyphal growth. *Nature*, 406:90–94, 2000.

[61] H. Zhu and M. Synder. Protein arrays and microarrays. *Curr. Opin. Chem. Biol.*, 5(1):40–5, 2001.