A Lanczos Procedure for Approximating Eigenvalues of Large Stochastic Matrices

by

William J. DeMeo

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Master of Science

in

Mathematics

in the

GRADUATE DIVISION
of the
NEW YORK UNIVERSITY

Committee in charge:

Professor Jonathan Goodman, Chair
Professor Leslie Greengard

January 1999

Professor Jonathan Goodman

To my parents, with love and appreciation.

Acknowledgments

First, I thank my advisor, Professor Jonathan Goodman, for giving me the opportunity to work on this problem, and helping me arrive at the following exposition. Next, I wish to thank Professor Helena Frydman for rst sparking my interest in Markov chains by giving lucid descriptions of their many interesting properties and applications. I thank Professors James Demmel and Beresford Parlett, for answering many questions pertaining to this problem, and Professor Leslie Greengard for agreeing to review this paper.

Finally, I would like to thank my family, for their support of my interests in research (and everything else!), and especially my parents, Barbara and Ted Terry, and Benita and Bill DeMeo. Needless to say, without them this paper would not have been written.

Abstract

A Lanczos Procedure for Approximating Eigenvalues of Large Stochastic Matrices

by

William J. DeMeo

Master of Science in Mathematics

New York University

Professor Jonathan Goodman, Chair

The rate at which a Markov chain converges to a given probability distribution has long been an active area of research. Well known bounds on this rate of convergence involve the subdominant eigenvalue of the chain's underlying transition probability matrix. However, many transition probability matrices are so large that we are unable to store even a vector of the matrix in fast computer memory. Thus, traditional methods for approximating eigenvalues are rendered useless.

In this paper we demonstrate that, if the Markov chain is reversible, and we understand the structure of the chain, we can derive the coefficients of the traditional Lanczos algorithm without storing a single vector. We do this by considering the variational properties of observables on the chain's state space. In the process we present the classical theory which relates the information contained in the Lanczos coefficients to the eigenvalues of the Markov chain.

Contents

Introduction

The rate at which a Markov chain converges to a given probability distribution has long been an active area of research. This is not surprising considering this problem's relevance to the areas of statistics, statistical mechanics, and computer science. Markov Chain Monte Carlo (MCMC) algorithms provide important examples. These algorithms come in handy when we encounter a complicated probability distribution from which we want to draw random samples. In statistical mechanics, we might wish to estimate the phase average of a function on the state space. Goodman and Sokal [6] examine Monte Carlo methods in this context. Examples from statistics occur in the Bayesian paradigm when we are forced to simulate an unwieldy posterior distribution (see, e.g., Geman and Geman [4].

To implement the MCMC algorithm, we invent a Markov chain that converges to the desired distribution (this is often accomplished using the *Metropolis algorithm* described in Chapter 4). Realizations of the chain will eventually represent samples from this distribution. Sometimes "eventually"—meaning all but finitely many terms of the chain—is just not enough. We need more practical results. In particular, we want to know how many terms of the chain should be discarded before we are sampling from a distribution that is close (in total variation distance) to the distribution of interest. This is the purpose of bounding convergence rates for Markov chains.

Often the Markov chains encountered in this context satisfy a condition known in the physics literature as *detailed balance*. Probabilists call chains with this property *reversible*. This simply means that the chain has the same probability law whether time moves forward or backward.[1] In this paper, we consider the

[1]This is not a precise definition. In particular the chain must have started from its stationary

rate at which such chains converge to a *stationary distribution*. This and other italicized terms are defined in Section 1.1.

There are a number of different methods in common use for bounding convergence rates of Markov chains, and a good review of these methods with many references can be found in [9]. More recently developed methods employing logarithmic Sobolev inequalities are reviewed in [2]. Most of the bounds in common use involve the subdominant eigenvalue of the Markov chain's transition probability matrix, and thus require good approximations to such eigenvalues. In many applications, however, the transition probability matrix is so large that it becomes impossible to store even a single vector of the matrix in conventional computer memory. These so called *out-of-core* problems are not amenable to traditional eigenvalue algorithms without modification.[2] In this paper we develop such a modification for the Markov chain eigenvalue problem. In particular we develop a method for approximating the first few eigenvalues of a transition probability matrix when we know the general structure of the underlying Markov chain. The method does not require storage of large matrices or vectors. Instead we need only simulate the Markov chain, and conduct a statistical analysis of the simulation.

Here is a brief summary of the paper. Section 1.1 contains a review of the relevant Markov chain theory. Readers who already know the basics of asymptotic theory of Markov chains might wish to skim Section1.1 if only to familiarize themselves with our notation. Section 1.2 describes functions on the state space of the Markov process. This section and Chapter 2 develop the context in which we formulate the new ideas of the paper. In the last section of Chapter 2, Sec-

distribution. Full rigor is postponed until Section 1.1.

[2]By "traditional eigenvalue algorithms" we mean those found, for example, in Golub and Van Loan [5]. See also the book by Demmel[1] for more recent treatment.

tion 2.2.1, we describe the *Krylov subspace* and explain why this represents our best approximation to a subspace containing extremal eigenvectors of the transition probability matrix (more precisely, a similarity transformation of this matrix). The first section of Chapter 3 describes the *Lanczos algorithm* for generating an orthonormal basis for the Krylov subspace. As it stands, this algorithm is useless for an out-of-core problem since, by definition of such problems, it requires too much data movement; all the computing time is spent swapping data between slow and fast memory (disk to ram to cache and back). We develop alternatives to the Lanczos algorithm and demonstrate that the *Lanczos coefficients* of the algorithm can be obtained by simulations of the Markov chain, and this allows us to avoid the standard algorithm altogether. Following this is a chapter describing the Metropolis algorithm used to produce a reversible stochastic matrix. It is here that we experiment with the procedure described in Section 3.2 and approximate the extremal eigenvalues of the matrix, without storing any of its vectors. Finally, Chapter 5 concludes the paper.

Part I

Theory

Chapter 1

Markov Chains

1.1 General Theory

This review of Markov chain theory can be found in any good probability text. The present discussion is most similar to that of Durrett [3], to which we refer the reader desiring greater detail.

1.1.1 The Basic Setup

Heuristically, a Markov chain is a stochastic process with a lack of memory property. Here this means that the future of the process, given its past behavior and its present state, depends only on its present state. This is the probabilistic analogue of a familiar property of classical particle systems. Given the position and velocities of all particles at time t, the equations of motion can be completely solved for the future evolution of the system. Thus, information describing the behavior of the process prior to time t is superuous. To be a bit more precise, if technical, we need the following definitions.

Definition 1.1. Let $(\mathcal{S}, \mathcal{S})$ be a measurable space. A sequence X_n, $n \geq 0$, of random variables taking values in $\mathcal{S}$ is said to be a Markov chain with respect to the filtration $\sigma(X_0, \ldots, X_n)$ if for all $B \in \mathcal{S}$,

$$P(X_{n+1} \in B \mid \sigma(X_0, \ldots, X_n)) = P(X_{n+1} \in B \mid \sigma(X_n)). \tag{1.1}$$

Equation (1.1) merely states that if we know the present location or state of X_n, then information about earlier locations or states is irrelevant for predicting X_{n+1}.

Definition 1.2. A function $p : S \times \mathcal{S} \to \mathbb{R}$ is said to be a *transition probability* if:

1. for each $x \in S$, $A \mapsto p(x, A)$ is a probability measure on $(S, \mathcal{S})$.
2. for each $A \in \mathcal{S}$, $x \mapsto p(x, A)$ is a measurable function.

We call X_n a Markov chain with transition probabilities p_n if

$$P(X_{n+1} \in B \mid \sigma(X_n)) = p_n(X_n, B) \tag{1.2}$$

The spaces $(S, \mathcal{S})$ that we encounter below are standard Borel spaces, so the existence of the transition probabilities follows from the existence of regular conditional probabilities on Borel spaces—a standard measure theory result (see e.g.[3]).

Suppose we are given an initial probability distribution μ on $(S, \mathcal{S})$ and a sequence p_n of transition probabilities. We can define a consistent set of finite dimensional distributions by

$$P(X_j \in B_j, 0 \leq j \leq n) = \int_{B_0} \mu(dx_0) \int_{B_1} p_0(x_0, dx_1) \cdots \int_{B_n} p_{n-1}(x_{n-1}, dx_n). \tag{1.3}$$

Furthermore, denote our probability space by

$$(\Omega, \mathcal{F}) = (S^\omega, \mathcal{S}^\omega), \quad \text{where } \omega = \{0, 1, \dots\}.$$

We call this *sequence space* and it is defined more explicitly by

$$S^\omega = \{(\omega_0, \omega_1, \dots) : \omega_i \in S\} \quad \text{and} \quad \mathcal{S}^\omega = \sigma(\omega : \omega_i \in A_i \in \mathcal{S}).$$

The Markov chain that we will study on this space is simply $X_n(\omega) = \omega$, the coordinate maps. Then, by the Kolmogorov extension theorem, there exists a unique probability measure P_μ on $(\Omega, \mathcal{F})$ so that the $X_n(\omega)$ have finite dimensional distributions (1.3).

If instead of μ we begin with the initial distribution δ_x, i.e., point mass at x, then we denote the probability measure by P_x. With such measures defined for each x, we can in turn define distributions P_μ, given any initial distribution μ, by

$$P_\mu(A) = \int \mu(dx) P_x(A).$$

That the foregoing construction—which, recall, was derived merely from an initial distribution μ and a sequence p_n of transition probabilities—satisfies Definition (1.2) of a Markov chain is not obvious, and a proof can be found in [3].

To state the converse of the foregoing, if X_n is a Markov chain with transition probabilities p_n and initial distribution μ, then its finite dimensional distributions are given by (1.3). Proof of this is also found in [3].

Now that we have put the theory on a firm, if abstract, foundation, we can bring the discussion down to earth by making the forgoing a little more concrete. First,

we specialize our study of Markov chains by assuming that our chain is *temporally homogeneous*, which means that the transition probabilities do not depend on time; i.e., $p_n(\omega_n, B) = p(\omega_n, B)$. (This is the stochastic analogue of a conservative system.) Next we assume that our state space S is finite, and suppose for all states $i, j \in S$ that $p(i, j) \geq 0$, and $\sum_j p(i, j) = 1$ for all i. In this case, equation (1.2) takes a more intuitive form,

$$P(X_{n+1} = j \mid X_n = i) = p(i, j),$$

and our transition probabilities become

$$p(i, A) = \sigma_{j \in A} p(i, j).$$

If P is a matrix whose (i, j) element is the transition probability $p(i, j)$ then P is a *stochastic matrix*; that is, a matrix with elements p_{ij} satisfying

$$p_{ij} \geq 0, \quad \sum_j p_{ij} = 1, \quad (i, j = 1, 2, \ldots, d).$$

We also refer to P as the transition probability matrix.

Without loss of generality, we can further suppose our Markov chain is *irreducible*. This means that, for any states i, j, starting in state i the chain will make a transition to state j at some future time with positive probability. This state of affairs is often described by saying that all states *communicate*. We lose no generality with this assumption because any *reducible* Markov chain can be factored into irreducible classes of states which can each be studied separately.

The final two conditions we place on the Markov chains considered below will

cost us some generality. Nonetheless, there remain many examples of chains meeting these conditions and making the present study worthwhile. Furthermore, it may be the case that, with a little more work, we will be able to drop these conditions in future studies. The first condition is that the chain is *aperiodic*. If we let $I_x = \{n \geq 1 : p^n(x,x) > 0\}$, we call a Markov chain *aperiodic* if, for any state x, the greatest common divisor of I_x is 1. The second assumption is that our chain is *reversible*. This characterization is understood in terms of the following definition.

Definition 1.3. A measure μ is called *reversible* if it satisfies

$$\mu(x)p(x,y) = \mu(y)p(y,x), \qquad \text{for all } x \text{ and } y.$$

We call a Markov chain *reversible* if its stationary distribution (defined in Section 1.1.2) is reversible.

1.1.2 A Convergence Theorem

In succeeding arguments, we use some results concerning the asymptotic behavior of Markov chains. These results require a few more definitions.

Definition 1.4. A measure π is said to be a *stationary measure* if

$$\sum_x \pi(x)p(x,y) = \pi(x). \tag{1.4}$$

Equation (1.4) says $P_\pi(X_1 = y) = \pi(y)$, and by induction that $P_\pi(X_n = y) = \pi(y)$ for all $n \geq 1$. If π is a probability measure, then we call π a stationary distribution. It represents an equilibrium for the chain in the sense that, if X_0 has distribution π, then so does X_n for all n.

When the Markov chain is irreducible and aperiodic, the distribution of the state at time n converges pointwise to π as $n \to \infty$, regardless of the initial state. It is convenient to state this convergence result in terms of the Markov chain's transition probability matrix P. Before doing so, we note that irreducibility of a Markov chain is equivalent to irreducibility (in the usual matrix theory sense) of its transition probability matrix. Furthermore, it turns out that a transition probability matrix of an aperiodic Markov chain falls into that class of matrices often called acyclic, but for simplicity we will call such stochastic matrices aperiodic. With this terminology, we can state the convergence theorem in terms of the transition probability matrix P.

Theorem 1.5. *Suppose* P *is irreducible, aperiodic, and has stationary distribution* π. *Then as* $n \to \infty$, $p^n(i,j) \to \pi(j)$.

The notation $p^n(i,j)$ means the (i,j) element of the nth power of P.

A Markov chain whose transition probability matrix satisfies the hypotheses of Theorem 1.5 is called *ergodic*. If we simulate an ergodic chain for sufficiently many steps, having begun in any initial state, the final state is a sample point from a distribution that is close to π.

To make this statement more precise requires that we define "close."

Definition 1.6. Let π be a probability measure on S. Then the *total variation distance* at time n with initial state x is given by

$$\Delta_x(n) = \|\mathrm{P}^n(x,A) - \pi(A)\|_{TV} = \max_{A \in \mathcal{S}} |\mathrm{P}^n(x,A) - \pi(A)|.$$

In what follows, we will measure rate of convergence using the function $\tau_x(\epsilon)$, defined as the first time after which the total variation distance is always less than

ϵ. That is,

$$\tau_x(\epsilon) = \min\{m : \Delta_x(n) \leq \epsilon \text{ for all } n \geq m\}.$$

To begin our consideration of the connection between convergence rates of Markov chains and eigenvalues, we first note that an aperiodic stochastic matrix P has real eigenvalues $1 = \lambda_0 > \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{d-1} \geq -1$, where $d = |S|$ is the dimension of the state space. For an ergodic chain, the rate of convergence to the stationary distribution π is bounded by a function of the *subdominant* eigenvalue. By subdominant eigenvalue we mean that eigenvalue which is second largest in absolute value, and we denote this eigenvalue by $\lambda_{\max} = \max \lambda_1, |\lambda_{d-1}|$. The function bounding the rate of convergence of a Markov chain is given by the following theorem (log denotes the natural logarithm):

Theorem 1.7. *The quantity $\tau_x(\epsilon)$ satisfies*

1. $\tau_x(\epsilon) \leq (1 - \lambda_{\max})^{-1}(\log \pi(x)^{-1} + \log \epsilon^{-1})$;

2. $\max_{x \in S} \tau_x(\epsilon) \geq \frac{1}{2}\lambda_{\max}(1 - \lambda_{\max})^{-1} \log(2\epsilon)^{-1}$.

As this theorem shows, if we have an upper bound on the subdominant eigenvalue, then we have an upper bound on the function $\tau_x(\epsilon)$. In what follows, we will derive an approximation to the subdominant eigenvalue and supply error bounds. Together, an approximation and error bounds for $\lambda_{\max}$ provide enough information to make Theorem 1.7 useful.

1.2 Functions on the State Space

Recall that $X_n(\omega) = \omega_n \in S$ denotes the state in which the Markov chain exists at time n. Suppose that $\Phi = \{\phi_1, \ldots, \phi_p\}$ is a collection of p *observables*, or functions

defined on the state space S. Furthermore, let these observables be real valued, $\phi_i : S \to \mathbb{R}$. It is often useful to assume that none of the observables is a constant function. Suppose now that the state space S is finite with d possible states. Then, since an observable is simply a map of the state space, we can think of each ϕ_i as a vector of d real numbers—the d values that it takes on at the different states.

Now assume the Markov chain is irreducible, and let π denote its stationary distribution. If we start the chain from its stationary distribution—i.e., suppose X_0 has distribution π—then X_n is a stationary process. Furthermore, for each i, $\phi_i(X_n)$ is a stationary stochastic process with *mean*

$$\mathbf{E}_\pi \phi_i = \sum_{x \in S} \pi(x)\phi_i(x)$$

and *autocovariance function*

$$\begin{aligned} \mathbf{C}_\pi(\phi_i(X_n), \phi_i(X_{n+s})) &= \mathbf{E}_\pi[(\phi_i(X_n) - \mathbf{E}_\pi\phi_i)(\phi_i(X_{n+s}) - \mathbf{E}_\pi\phi_i)] \\ &= \sum_{x,y \in S} P_\pi(X_n = x, X_{n+s} = y)(\phi_i(x) - \mathbf{E}_\pi\phi_i)(\phi_i(y) - \mathbf{E}_\pi\phi_i). \end{aligned} \tag{1.5}$$

By the definition of conditional probability, we can write (1.4) as follows:

$$\sum_{x,y \in S} P_\pi(X_n = x) P_\pi(X_{n+s} = y \mid X_n = x)(\phi_i(x) - \mathbf{E}_\pi\phi_i)(\phi_i(y) - \mathbf{E}_\pi\phi_i).$$

Equivalently,

$$\sum_{x,y \in S} \pi(x) p^s_{xy} (\phi_i(x) - \mathbf{E}_\pi\phi_i)(\phi_i(y) - \mathbf{E}_\pi\phi_i).$$

Here p^s_{xy} denotes the element in row x and column y of P^s, the sth power of the transition probability matrix. Similarly, we define the *cross-covariance* between

the function ϕ_i at time n and ϕ_j at time $n+s$ as

$$\begin{aligned}\mathbf{C}_\pi(\phi_i(X_n),\phi_j(X_{n+s})) &= \mathbf{E}_\pi[(\phi_i(X_n)-\mathbf{E}_\pi\phi_i)(\phi_j(X_{n+s})-\mathbf{E}_\pi\phi_j)] \\ &= \sum_{x,y\in S}\pi(x)p_{xy}^s(\phi_i(x)-\mathbf{E}_\pi\phi_i)(\phi_j(y)-\mathbf{E}_\pi\phi_j). \qquad (1.6)\end{aligned}$$

Now let $\langle\Phi\rangle$ denote the matrix of mean vectors whose jth column is $\mathbf{E}_\pi\phi_j\mathbf{1}$, where $\mathbf{1}=(1,\ldots,1)^t$, and let $\Pi=\mathrm{diag}(\pi(\omega_1),\ldots,\pi(\omega_d))$ be the $d\times d$ diagonal matrix with stationary probabilities $\pi(\omega)$ on the main diagonal and zeros elsewhere. Finally, denoting by $\mathrm{C}(s)$ the $p\times p$ covariance matrix whose (i,j) element is $\mathbf{C}_\pi(\phi_i(X_n),\phi_j(X_{n+s}))$, we have

$$\begin{aligned}\mathrm{C}(0) &= \mathrm{E}(\Phi(X_n)-\langle\Phi\rangle)(\Phi(X_n)-\langle\Phi\rangle)^t \\ &= (\Phi-\langle\Phi\rangle)^t\Pi(\Phi-\langle\Phi\rangle), \\ \mathrm{C}(s) &= \mathrm{E}(\Phi(X_n)-\langle\Phi\rangle)(\Phi(X_{n+s})-\langle\Phi\rangle)^t \\ &= (\Phi-\langle\Phi\rangle)^t\Pi\mathrm{P}^s(\Phi-\langle\Phi\rangle).\end{aligned}$$

Below, we will also find it useful to have at our disposal a new matrix that is *similar* to the transition probability matrix. We have in mind the matrix $\mathrm{M}=\Pi^{1/2}\mathrm{P}\Pi^{-1/2}$. As is easily verified, this allows us to write the covariance matrix as

$$\begin{aligned}\mathrm{C}(s) &= (\Phi-\langle\Phi\rangle)^t\Pi^{\frac{1}{2}t}\mathrm{M}^s\Pi^{\frac{1}{2}}(\Phi-\langle\Phi\rangle) \\ &= \Psi^t\mathrm{M}^s\Psi, \qquad (1.7)\end{aligned}$$

where we have defined $\Psi=\Pi^{\frac{1}{2}}(\Phi-\langle\Phi\rangle)$. Recall that our main motivation is that for out-of-core problems traditional eigenvalue algorithms are inadequate. With

this fact and the form (1.7) in mind, we will consider using the covariance of observables on the state space to implement the Rayleigh-Ritz procedure, which we describe below. This procedure requires that M be symmetric. As the next fact demonstrates, this need for symmetry is the reason we insist that the Markov chain be reversible.

Fact. The matrix M is symmetric if and only if the Markov chain is reversible (i.e., iff the process satisfies the *detailed balance* condition).

Proof.

$$\begin{aligned} \text{M is symmetric} \quad &\iff \quad (\Pi^{1/2}\mathrm{P}\Pi^{-1/2})^t = \Pi^{1/2}\mathrm{P}\Pi^{-1/2} \\ &\iff \quad \Pi^{-\frac{1}{2}t}\mathrm{P}^t\Pi^{\frac{1}{2}t} = \Pi^{1/2}\mathrm{P}\Pi^{-1/2} \\ &\iff \quad \mathrm{P}^t\Pi^t = \Pi\mathrm{P}. \end{aligned}$$

Elementwise, the final equality is $\pi_i p_{ij} = \pi_j p_{ji}$. According to Definition 1.3, this states that p is a reversible measure. □

Chapter 2

Invariant and Approximate Invariant Subspaces

2.1 Invariant Subspaces

Definition 2.1. A subspace $S \subseteq \mathbb{R}^n$ with the property that

$$x \in S \quad \Longrightarrow \quad \mathrm{M}x \in S$$

is said to be *invariant* for M.

Recall that, having chosen observables $\langle \Phi \rangle = (\phi_1, \ldots, \phi_p)$, we constructed the covariance matrix $\mathrm{C}(s) = \Psi^t \mathrm{M} \Psi$. If the column space of Ψ, which we denote by $\mathrm{ran}(\Psi)$, is an invariant subspace for M, the definition implies $\mathrm{M}\psi_j \in \mathrm{ran}(\Psi)$. That is, for each ψ_i there exists a vector t of coefficients such that $\mathrm{M}\psi_j = \sum_{i=1}^{p} t_i \psi_{ij}$. This is true for all j and, putting each vector of coefficients into a matrix T, we see that $\mathrm{M}\Psi = \Psi \mathrm{T}$. Conversely, $\mathrm{M}\Psi = \Psi \mathrm{T}$ implies that $\mathrm{M}\psi_j$ is a linear combination

of columns of Ψ, so ran(Ψ) is invariant. We have thus proved the following

Fact. The subspace ran(Ψ) is invariant for M if and only if there exists $\mathrm{T} \in \mathbb{R}^{p \times p}$ such that $\mathrm{M}\Psi = \Psi\mathrm{T}$.

Consequently,

Fact. $\lambda(\mathrm{T}) \subseteq \lambda(\mathrm{M})$.

Proof.

$$\begin{aligned} \lambda \in \lambda(\mathrm{T}) &\iff (\exists v \in \mathbb{R}^p)\mathrm{T}v = \lambda v \\ &\iff \Psi\mathrm{T}v = \lambda\Psi v \\ &\iff \mathrm{M}\Psi v = \lambda\Psi v \\ &\iff \lambda \in \lambda(M). \end{aligned}$$

The second equivalence follows from Fact 2.1. □

To see why Facts 2.1 and 2.1 are theoretically useful, note that the equation of Fact 2.1, $\Psi\mathrm{T} = \mathrm{M}\Psi$, is equivalent to $\Psi^t\Psi\mathrm{T} = \Psi^t\mathrm{M}\Psi$, which is equivalent to $\mathrm{T} = (\Psi^t\Psi)^t\Psi^t\mathrm{M}\Psi$. In this form, we recognize that $\mathrm{T} = \mathrm{C}^{-1}(0)\mathrm{C}(1)$. That is, using only the covariance of observables on the state space, we can generate a matrix T with the property $\lambda(\mathrm{T}) \subseteq \lambda(\mathrm{M})$. Recalling that $\mathrm{M} = \Pi^{1/2}\mathrm{P}\Pi^{-1/2}$, we see that M is similar to our original stochastic matrix P, and thus $\lambda(M) = \lambda(P)$.

2.2 Approximate Invariant Subspaces

We noted above that Facts 2.1 and 2.1 are theoretically useful. Speaking practically now, when choosing observables on the state space we may not be sure that

they will satisfy the primary assumption underlying the two facts. Recall the assumption: $\mathrm{ran}(\Psi)$ is an invariant subspace for M where $\Psi = \Pi^{1/2}(\Phi - \langle\Phi\rangle)$. It may well be the case that there exists $\psi_j \in \mathrm{ran}(\Psi)$ such that $\mathrm{M}\psi_j \notin \mathrm{ran}(\Psi)$, thereby violating the assumption. Even if we are lacking an invariant subspace, however, for some applications it is reasonable to expect that observables can be chosen to provide at least an *approximate invariant subspace*, which is defined as follows:

Definition 2.2. If the columns of $\Psi \in \mathbb{R}^{d\times p}$ are independent and the norm of the *residual matrix* $\mathrm{E} = \mathrm{M}\Psi - \Psi\mathrm{T}$ is small for some $\mathrm{T} \in \mathbb{R}^{p\times p}$, then $\mathrm{ran}(\Psi)$ defines an *approximate invariant subspace*.

To see how an approximate invariant subspace can be useful for approximating eigenvalues of M, we recall a theorem from Golub and Van Loan [5].

Theorem 2.3. *Suppose* $\mathrm{M} \in R^{d\times d}$ *and* $\mathrm{T} \in \mathbb{R}^{p\times p}$ *are symmetric and let* $\mathrm{E} = \mathrm{MQ} - \mathrm{QT}$*, where* $\mathrm{Q} \in \mathbb{R}^{d\times p}$ *is orthonormal (i.e.,* $\mathrm{Q}^t\mathrm{Q} = \mathrm{I}$*). Then there exist* $\mu_1, \ldots, \mu_p \in \lambda(\mathrm{T})$ *and* $\lambda_1, \ldots, \lambda_p \in \lambda(\mathrm{M})$ *such that*

$$|\mu_k - \lambda_k| \leq \sqrt{2}\|\mathrm{E}\|_2, \qquad \textit{for } k = 1, \ldots, p.$$

If the subspace $\mathrm{ran}(\mathrm{Q})$ is an approximate invariant subspace, then the definition implies that there is a choice T rendering the error $\|\mathrm{E}\|_2$ small, and thus the eigenvalues of T provide a good approximation to those of M.

When considering the foregoing ideas, it is apparent that their application presents new—but hopefully less prohibitive—problems. As these problems are the focus of the rest of the paper, now is a good time to examine them.

First, the preceding theorem assumes a matrix Q whose columns form an orthonormal basis for the approximate invariant subspace. For our problem, deriva-

tion of such a Q is tricky, and we must prepare for this.

Next, having an approximate invariant subspace at our disposal merely tells us that there exists some matrix T which makes the error $\|\mathrm{E}\|_2$ small. We must discover the form of such a T. Moreover, it is natural to seek that T which minimizes $\|\mathrm{E}\|_2$ for a given approximate invariant subspace.

Finally, in order to apply these ideas to realistic eigenvalue problems, we must find a practical way to generate a good approximate invariant subspace.

We now address each of these issues in turn.

Recall the matrix $\Psi = \Pi^{1/2}(\Phi\langle\Phi\rangle)$. The columns of this matrix, though independent (by choice of independent observables), are not necessarily orthonormal. However, consider the polar decomposition $\Psi = \mathrm{QZ}$, where $\mathrm{Q}^t\mathrm{Q} = \mathrm{I}$ and $\mathrm{Z}^2 = \Psi^t\Psi$ is a symmetric positive semidefinite matrix.[1] Note that $\mathrm{Z} = (\Psi^t\Psi)^{1/2}$ is nonsingular, so Q has the form

$$\mathrm{Q} = \Psi\mathrm{Z}^{-1} = \Psi(\Psi^t\Psi)^{-1/2},$$

and it is clear that $\mathrm{ran}(\mathrm{Q}) = \mathrm{ran}(\Psi)$. Perhaps $\mathrm{ran}(\mathrm{Q})$ is a useful approximation to the invariant subspace for M. If $\mathrm{ran}(Q)$ is not itself invariant, we have the error matrix $\mathrm{E} = \mathrm{MQ} - \mathrm{QT}$. Below we show that the T which minimizes $\|\mathrm{E}\|_2$ is $\mathrm{T} = \mathrm{Q}^t\mathrm{MQ}$. This yields the following

Theorem 2.4. *If $\Psi = \mathrm{QZ}$ is the polar decomposition of Ψ, then the matrix*

$$\mathrm{T} = \mathrm{Q}^t\mathrm{MQ} = (\Psi^t\Psi)^{-1/2}\Psi^t\mathrm{M}^t\Psi(\Psi^t\Psi)^{-1/2}$$

minimizes $\|\mathrm{E}\|_2 = \|\mathrm{MQ} - \mathrm{QT}\|_2$.

[1] Recall that the polar decomposition is derived from the singular value decomposition, $\Psi = \mathrm{U}\Sigma\mathrm{V}^t$, by letting $\mathrm{Q} = \mathrm{UV}^t$ and $\mathrm{Z} = \mathrm{V}\Sigma\mathrm{V}^t$.

Proof. We prove the result by establishing the following

Claim: Given $\mathrm{M} \in \mathbb{R}^{d\times d}$ suppose $Q \in \mathbb{R}^{d\times d}$ satisfies $\mathrm{Q}^t\mathrm{Q} = \mathrm{I}$. Then,

$$\min_{\mathrm{T}\in\mathbb{R}^{p\times p}} \|\mathrm{MQ} - \mathrm{QT}\|_2 = \|(\mathrm{I} - \mathrm{QQ}^t)\mathrm{MQ}\|_2,$$

and $\mathrm{T} = \mathrm{Q}^t\mathrm{MQ}$ is the minimizer. The claim is verified by an easy application of the Pythagorean theorem: For any $\mathrm{T} \in \mathbb{R}^{p\times p}$ we have

$$\begin{aligned}
\|\mathrm{MQ} - \mathrm{QT}\|_2^2 &= \|\mathrm{MQ} - \mathrm{QQ}^t\mathrm{MQ} + \mathrm{QQ}^t\mathrm{MQ} - \mathrm{QT}\|_2^2 \\
&= \|(\mathrm{I} - \mathrm{QQ}^t)\mathrm{MQ} + \mathrm{Q}(\mathrm{Q}^t\mathrm{MQ} - \mathrm{T})\|_2^2 \\
&= \|(\mathrm{I} - \mathrm{QQ}^t)\mathrm{MQ}\|_2^2 + \|\mathrm{Q}(\mathrm{Q}^t\mathrm{MQ} - \mathrm{T})\|_2^2 \qquad (2.1) \\
&= \|(\mathrm{I} - \mathrm{QQ}^t)\mathrm{MQ}\|_2^2
\end{aligned}$$

Equality (2.1) holds since $\mathrm{I} - \mathrm{QQ}^t$ projects MQ onto the subspace orthogonal to $\mathrm{ran}(Q)$. Thus, the two terms in the expression on the right are orthogonal, and the Pythagorean theorem yields equality. The concluding inequality establishes that the minimizing T is that which annihilates the second term in (2.1), that is, $\mathrm{T} = \mathrm{Q}^t\mathrm{MQ}$.

The second equality in Theorem 2.4 is a consequence of the polar decomposition, in which $\mathrm{Q} = \Psi\mathrm{Z}^{-1} = \Psi(\Psi^t\Psi)^{-1/2}$. Thus, $\mathrm{T} = (\Psi^t\Psi)^{-1/2}\Psi^t\mathrm{M}^t\Psi(\Psi^t\Psi)^{-1/2}$ is the minimizer, as claimed. □

2.2.1 The Krylov Subspace

Suppose the columns of a matrix $\mathrm{Q} \in \mathbb{R}^{d\times p}$ give an orthonormal basis for an approximate invariant subspace. Then, as we have seen,

1. $\mathrm{T} = \mathrm{Q}^t\mathrm{MQ}$ minimizes $\|\mathrm{E}\|_2 = \|\mathrm{MQ} - \mathrm{QT}\|_2$ and

2. there exist $\mu_1, \ldots, \mu_p \in \lambda(\mathrm{T})$ and $\lambda_1, \ldots, \lambda_p \in \lambda(\mathrm{M})$ such that

$$\|\mu_k - \lambda_k\| \leq \sqrt{2}\|\mathrm{E}\|_2, \quad \text{for } k = 1, \ldots, p.$$

Given an approximate invariant subspace ran(Q) of dimension p, these facts tell us what matrix we should use to approximate p elements of the spectrum of M. Now all we lack is a description of ran(Q). That is, we have not specified which approximate invariant subspace would best suit our objective of approximating the subdominant eigenvalue $\lambda_{\max}(\mathrm{M})$. For this purpose the following definition is useful:

Definition 2.5. The *Raleigh quotient* of a symmetric matrix M and a nonzero vector x is

$$\rho(x, \mathrm{M}) = \frac{x^t\mathrm{M}x}{x^t x}.$$

We will denote Raleigh quotient by $\rho(x)$ when the context makes clear what matrix is involved.

To find the approximate invariant subspace most appropriate for our problem, we choose each dimension successively, providing justification at each step. We start with one observable ϕ on the state space, and let $\psi_1 = \Pi^{1/2}(\phi - \mathrm{E}_\pi\phi)$. That is, ψ_1 is a centered (mean zero) observable whose ith coordinate is weighted by $\sqrt{\pi(i)}$. Notice that the definition (which comes from the definition of Ψ following Equation (1.7)) is such that ψ_1 is a constant vector if and only if ϕ is constant on the state space, in which case ψ_1 is the constant zero function. (Observables that are constant on the state space are not interesting.) Second, recall that the row

sums of the matrix P are all one, therefore the eigenvector corresponding to the eigenvalue $\lambda_0(\mathrm{P}) = 1$ is the constant vector. By definition of $\mathrm{M} = \Pi^{1/2}\mathrm{P}\Pi^{-1/2}$, we see that the constant vector is also the eigenvector corresponding to λ_0. This will play an important role in what follows, as it allows us to focus on the subdominant eigenvalue $\lambda_{\max}(\mathrm{M}) = \max\{\lambda_1(\mathrm{M}), |\lambda_{d-1}(\mathrm{M})|\}$ rather than on $\lambda_0(M)$ (which we already know is 1).

Now, notice that

$$|\rho(\psi_1, \mathrm{M})| \leq \max |\rho(x, \mathrm{M})| = \lambda_{\max}(\mathrm{M}) \tag{2.2}$$

where the max is taken over all nonconstant vectors. Since our interest centers on $\lambda_{\max}(\mathrm{M})$, we would like a ψ_1 that makes the left hand side of (2.2) large. This would be achieved if ψ_1 were to lie in the space spanned by, say, the first two eigenvectors of the Markov chain (sometimes referred to as the *slowest modes* of the process). However, this subspace spans only two dimensions of the entire d-dimensional space, and it is more likely that ψ_1 only comes close, at best, to lying in the subspace of interest. Now, given ψ_1, a judicious choice for the second dimension ψ_2, and hence $\Psi_2 = [\psi_1, \psi_2]$, would be that which makes $\max_{a \neq 0} |\rho(\Psi_2 a)|$ as large as possible. To establish that this is indeed the right objective, note the following:

$$\begin{aligned}
\max_{a \neq 0} |\rho(\Psi_2 a, \mathrm{M})| &= \max_{a \neq 0} \left| \frac{a^t \Psi_2^t \mathrm{M} \Psi_2 a}{a^t \Psi_2^t \Psi_2 a} \right| \\
&= \max_{x \in \mathrm{ran}(\Psi_2)} |\rho(x, \mathrm{M})| \\
&= \max |\rho(x, \mathrm{M})| \\
&= \lambda_{\max}(\mathrm{M})
\end{aligned} \tag{2.3}$$

Again, the max on the right side of (2.3) is over nonconstant vectors. In other words, we wish to chose $\Psi_2 = [\psi_1, \psi_2]$ so that there is a vector $a \in \mathbb{R}^2$ making $|\rho(\Psi_2 a, \mathrm{M})|$ close to $\lambda_{\max}(\mathrm{M})$.

Now, $\rho(\psi_1)$ changes most rapidly in the direction of the gradient $\nabla\rho(\psi_1)$.

$$\nabla\rho(\psi_1) = \left(\frac{\partial\rho(\psi_1)}{\partial\psi_1(1)}, \ldots, \frac{\partial\rho(\psi_1)}{\partial\psi_1(d)}\right) = \frac{2}{\psi_1^t\psi_1}(\mathrm{M}\psi_1 - \rho(\psi_1)\psi_1). \tag{2.4}$$

So, to maximize the left hand side of (2.3), Ψ_2 should be chosen so that the subspace $\mathrm{ran}(\Psi_2)$ contains the gradient vector. That is, we must choose ψ_2 so that

$$\nabla\rho(\psi_1) \in \mathrm{ran}\{\psi_1, \psi_2\} = \mathrm{ran}(\Psi_2). \tag{2.5}$$

Clearly, Equation (2.4) implies $\nabla\rho(\psi_1) \in \mathrm{ran}\{\psi_1, \mathrm{M}\psi_1\}$. Thus, if $\mathrm{ran}\{\psi_1, \psi_2\} = \mathrm{ran}\{\psi_1, \mathrm{M}\psi_1\}$, then (2.5) is satisfied. In general, having chosen $\Psi_k = [\psi_1, \ldots, \psi_k]$ so that $\mathrm{ran}\{\psi_1, \psi_2, \ldots, \psi_k\} = \mathrm{ran}\{\psi_1, \mathrm{M}\psi_1, \ldots, \mathrm{M}\psi_k\}$, we must chose ψ_{k+1} so that for any nonzero vector $a \in \mathbb{R}^k$,

$$\nabla\rho(\Psi_k a) \in \mathrm{ran}\{\psi_1, \psi_2, \ldots, \psi_k\}. \tag{2.6}$$

Now,

$$\nabla\rho(\Psi_k a) = \frac{2}{a^t\Psi_k^t\Psi_k a}(\mathrm{M}\Psi_k - \rho(\Psi_k a)\Psi_k a).$$

and therefore,

$$\nabla\rho(\Psi_k a) \in \mathrm{ran}(\mathrm{M}\Psi_k) \cup \mathrm{ran}(\Psi_k) = \mathrm{ran}\{\psi_1, \mathrm{M}\psi_1, \ldots, \mathrm{M}^k\psi_1\}.$$

Thus, the requirement (2.6) is satisfied when

$$\mathrm{ran}\{\psi_1, \psi_2, \ldots, \psi_k\} = \mathrm{ran}\{\psi_1, \mathrm{M}\psi_1, \ldots, \mathrm{M}^k\psi_1\}.$$

In conclusion, the p-dimensional approximate invariant subspace that is most suitable for our objective is

$$\mathcal{K}(\mathrm{M}, \psi_1, p) = \mathrm{ran}\{\psi_1, \mathrm{M}\psi_1, \ldots, \mathrm{M}^{p-1}\psi_1\}.$$

This is known as the *Krylov subspace*. Therefore, to answer the problem posed at the outset of this section, if we take the columns of Q to be an orthonormal basis for $\mathcal{K}(\mathrm{M}, \psi_1, p)$, then the eigenvalues of $\mathrm{T} = \mathrm{Q}^t\mathrm{MQ}$ should provide good estimates of p extremal eigenvalues of M.

Chapter 3

Lanczos Procedures

The following will review the usual procedure for generating an orthonormal basis for $\mathcal{K}(\mathrm{M}, \psi_1, p)$, and conclude by showing that, for our problem, we can find the matrix $\mathrm{Q}^t\mathrm{MQ}$ without actually carrying out this procedure. Note that this conclusion is essential since the procedure requires matrix vector multiplication—an operation we have assumed impossible for large enough d.

3.1 The General Lanczos Algorithm

Let $\psi^{(k)} = \mathrm{M}^k\psi$ and consider the $d \times d$ matrix

$$\Psi = [\psi, \mathrm{M}\psi, \ldots, \mathrm{M}^{d-1}\psi] = [\psi^{(0)}, \psi^{(1)}, \ldots, \psi^{(d-1)}]$$

Notice that $\mathrm{M}\Psi = [\psi^{(1)}, \psi^{(2)}, \ldots, \psi^{(d-1)}, \mathrm{M}^d\psi]$, and assuming for the moment that Ψ is nonsingular, we can compute the vector $h = -\Psi^{-1}\mathrm{M}^d\psi$. Thus,

$$\mathrm{M}\Psi = \Psi[e_2, \ldots, e_d, -h], \tag{3.1}$$

where e_j is the column vector with 1 in the jth position and zeros elsewhere. We define $H = [e_2, \ldots, e_d, -h]$, so (3.1) becomes $\mathrm{M}\Psi = \Psi H$. Equivalently,

$$H = \begin{pmatrix} 0 & 0 & \cdots & 0 & -h_1 \\ 1 & 0 & \cdots & 0 & -h_2 \\ & \ddots & & & \vdots \\ & & \ddots & 0 & -h_{d-1} \\ & & & 1 & -h_d \end{pmatrix} = \Psi^{-1}\mathrm{M}\Psi. \tag{3.2}$$

H is a *companion matrix* which means that its characteristic polynomial is $p(x) = x^d + \sum_{i=1}^{d} h_i x^{i-1}$. Since H is similar to M, finding the eigenvalues of M is equivalent to finding the roots of $p(x)$. However, this is of little practical use since finding h, constructing $p(x)$, and finding its roots is probably a harder problem than the one we started with. Instead, the value of decomposition (4.1) derives from its *upper Hessenberg form*. We exploit this property below.

Let $\Psi = \mathrm{QR}$ be the QR decomposition of Ψ. Since Ψ is assumed nonsingular,

$$\Psi_d^{-1}\mathrm{M}\Psi_d = (\mathrm{R}^{-1}\mathrm{Q}^t)\mathrm{M}(\mathrm{QR}) = \mathrm{H}$$

Therefore, $\mathrm{Q}^t\mathrm{MQ} = \mathrm{RHR}^{-1}$. Let TRHR^{-1}. Then, since R and R^{-1} are both upper triangular and H is upper Hessenberg $\mathrm{T} = \mathrm{RHR}^{-1}$ is also upper Hessenberg. Furthermore, since M is symmetric, it is clear that $\mathrm{T}^t = \mathrm{Q}^t\mathrm{MQ} = T$. Thus, T is both upper Hessenberg and symmetric. Therefore, T is tridiagonal and we can

write it as follows:

$$\mathrm{T} = \begin{pmatrix} \alpha_1 & \beta_1 & 0 & \cdots & 0 \\ \beta_1 & \alpha_2 & \ddots & & \vdots \\ & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & \beta_{d-1} \\ 0 & \cdots & & \beta_{d-1} & \alpha_d \end{pmatrix}. \tag{3.3}$$

Equating columns j on both sides of $\mathrm{MQ} = \mathrm{QT}$ yields

$$\mathrm{M}q_j = \beta_{j-1}q_{j-1} + \alpha_j q_j + \beta_j q_{j+1} \tag{3.4}$$

Since the columns of Q are orthonormal, multiplying both sides of (3.4) by q_j and q_{j+1} yields $\alpha_j = q_j^t \mathrm{M} q_j$ and $\beta_j = q_{j+1}^t \mathrm{M} q_j$. The foregoing justifies what is called the *Lanczos algorithm*, which is performed as follows: The q_j computed by the

Algorithm 1 The Lanczos algorithm for partial reduction to symmetric tridiagonal form.

```
q_1 = ψ/‖ψ‖_2, β_0 = 0, q_0 = 0
for j = 1, ..., p do
   z ← Mq_j
   α_j ← q_j^t z
   z ← z − α_j q_j − β_{j−1} q_{j−1}
   β_j ← ‖z‖_2
   if β_j = 0 then
      return
   end if
   q_{j+1} = z/β_j
end for
```

Lanczos algorithm are often called the *Lanczos vectors*. If the loop in the algorithm is terminated because $\beta_k = 0$, this indicates that an exact invariant subspace has been computed, and is given by $\mathrm{ran}\{q_1, \ldots, q_k\}$. Otherwise, we usually halt the algorithm after p steps, in which case the algorithm converges to approximations

of at most p eigenvalues.[1]

Note that Equation (3.4) results from the equation MQ = QT. The latter equation holds since, starting with d vectors in our subspace, Q is a $d \times d$ matrix whose columns form an orthonormal basis for all of $\mathbb{R}^d$, which is clearly an invariant subspace. The Lanczos algorithm above, however, proceeds for only p steps, producing a $d \times p$ matrix Q, whose columns form an orthonormal basis for the *approximate* invariant subspace $\mathcal{K}(\mathrm{M}, \psi, p)$, and a $p \times p$ matrix $\mathrm{T}, = \mathrm{Q}^t\mathrm{MQ}$. In that case, $\|\mathrm{MQ} - \mathrm{QT}_p\|_2 = \|\mathrm{E}\|_2$ is nonzero and gives the error bound described in Theorem 2.3. Now, writing the full $d \times d$ matrix T of (3.3) as

$$\mathrm{T} = \left(\begin{array}{c|c} \mathrm{T}_p & \mathrm{T}_{pu}^t \\ \hline \mathrm{T}_{pu} & \mathrm{T}_u \end{array}\right) = \left(\begin{array}{cccc|ccccc} \alpha_1 & \beta_1 & & & & & & & \\ \beta_1 & \ddots & \ddots & & & & & & \\ & \ddots & \ddots & \beta_{p-1} & & & & & \\ & & \beta_{p-1} & \alpha_p & \beta_p & & & & \\ \hline & & & \beta_p & \alpha_{p+1} & \beta_{p+1} & & & \\ & & & & \beta_{p+1} & \ddots & \ddots & & \\ & & & & & \ddots & \ddots & \beta_{d-1} & \\ & & & & & & \beta_{d-1} & \alpha_d & \end{array}\right) \tag{3.5}$$

allows us to describe the error bound $\|\mathrm{E}\|_2$ in terms of the submatrix T_{pu}, and hence in terms of β_p.

Theorem 3.1. *If* T_p *and* T_{pu} *are the matrices appearing in (3.5), and if the* p *columns of* Q *are computed by the Lanczos algorithm, then there exist* $\mu_1, \ldots, \mu_p \in$

[1] A Detailed discussion of such convergence (and misconvergence) is given in [1].

$\lambda(\mathrm{T}_p)$ *and* $\lambda_1, \ldots, \lambda_p \in \lambda(\mathrm{M})$ *such that*

$$|\mu_k - \lambda_k| \leq \|\mathrm{T}_{pu}\|_2 = \beta_p, \qquad \textit{for } k = 1, \ldots, p.$$

For a proof, see [1, Page 365].

3.2 A Lanczos Procedure for Markov Chains

Notice that the Lanczos algorithm requires the computation of $\mathrm{M}q_j$. Throughout this paper we have assumed that $\mathrm{M} \in \mathbb{R}^{d\times d}$, and that d is so large as to make the matrix vector multiplication $\mathrm{M}q_j$ impossible. Now, the Lanczos algorithm constructs an orthonormal basis for the Krylov subspace. For our purposes, this construction is not essential. What is essential is that we find a way to generate the Lanczos coefficients α_i, β_j, $(j = 1, \ldots, p)$, and whence the matrix T, without performing the operation $\mathrm{M}q_j$ required by the Lanczos algorithm. We now address this problem.

Begin with the centered and weighted observable $\psi = \Pi^{1/2}(\phi - \mathbf{E}_\pi\phi)$, and let $q_1 = \psi/\|\psi\|_2$. To simplify notation, let $\phi(n) = \phi(X_n)$. The form of the first Lanczos coefficient α_1 is straight forward:

$$\alpha_1 = q_1^t \mathrm{M} q_1 = \frac{\psi^t \mathrm{M}\psi}{\psi^t\psi}.$$

and now the special properties of the Markov chain setting play an important role.

Indeed, we have

$$\begin{aligned}\psi^t \mathrm{M}\psi &= (\phi - \mathbf{E}_\pi\phi)^t \Pi^{\frac{1}{2}t} \mathrm{M} \Pi^{\frac{1}{2}} (\phi - \mathbf{E}_\pi\phi) \\ &= \mathbf{C}_\pi(\phi(n), \phi(n+1)). \end{aligned} \tag{3.6}$$

and

$$\begin{aligned}\psi^t \psi &= (\phi - \mathbf{E}_\pi\phi)^t \Pi (\phi - \mathbf{E}_\pi\phi) \\ &= \mathbf{Var}_\pi(\phi(n)). \end{aligned} \tag{3.7}$$

Recall that the definition of M implies $\Pi^{\frac{1}{2}t} \mathrm{M} \Pi^{\frac{1}{2}} = \Pi \mathrm{P}$, and this justifies Equation (3.6). Putting these equations together, we have a nice form for α_1:

$$\alpha_1 = \frac{\mathbf{C}_\pi(\phi(n), \phi(n+1))}{\mathbf{Var}_\pi(\phi(n))}.$$

The next Lanczos coefficient, β_1, is only slightly more complicated.

$$\begin{aligned}\beta_1 &= \|\mathrm{M}q_1 - \alpha_1 q_1\|_2 \\ &= \left[(\mathrm{M}q_1 - \alpha_1 q_1)^t (\mathrm{M}q_1 - \alpha_1 q_1)\right]^{1/2} \\ &= \left[q_1^t \mathrm{M}^2 q_1 - \alpha_1^2\right]^{1/2}. \end{aligned} \tag{3.8}$$

Again recalling the definition $\mathrm{M} = \Pi^{1/2} \mathrm{P} \Pi^{1/2}$, a simple manipulation verifies

$\Pi^{\frac{1}{2}t}\mathrm{M}^s\Pi^{\frac{1}{2}} = \Pi\mathrm{P}^s$, whereby it follows that

$$\begin{aligned} q_1^t\mathrm{M}^2 q_1 &= \frac{(\phi - \mathbf{E}_\pi\phi)^t\Pi\mathrm{P}^2(\phi - \mathbf{E}_\pi\phi)}{(\phi - \mathbf{E}_\pi\phi)^t\Pi(\phi - \mathbf{E}_\pi\phi)} \\ &= \frac{\mathbf{C}_\pi(\phi(n), \phi(n+2))}{\mathbf{Var}_\pi(\phi(n))}. \end{aligned}$$

Inserting the last expression into Equation (3.8) gives

$$\beta_1 = \left[\frac{\mathbf{C}_\pi(\phi(n), \phi(n+2))}{\mathbf{Var}_\pi(\phi(n))} - \left(\frac{\mathbf{C}_\pi(\phi(n), \phi(n+1))}{\mathbf{Var}_\pi(\phi(n))}\right)^2\right]^{1/2}.$$

Notice that we can estimate these expressions of α_1 and β_1 by running simulations of the Markov chain. To do so, we run the Markov chain starting from a random initial state, compute the value of the observable at each step of the chain and, from the data, compute estimates of the covariances and variance of the observable.

Before generalizing this for the jth Lanczos coefficient, we consider one more coefficient in detail. Recall that the second basis element q_2 in the Lanczos algorithm is given by $q_2 = (\mathrm{M}q_1\alpha_1 q_1)/\beta_1$. From this we can derive an expression for α_2 which, while not as neat as those for α_1 and β_1, is nonetheless tractable.

$$\begin{aligned} \alpha_2 &= q_2^t\mathrm{M}q_2 \\ &= \frac{1}{\beta_1^2}(\mathrm{M}q_1 - \alpha_1 q_1)^t\mathrm{M}(\mathrm{M}q_1 - \alpha_1 q_1) \\ &= \frac{1}{\beta_1^2}(q_1^t\mathrm{M}^3 q_1 - 2\alpha_1 q_1^t\mathrm{M}^2 q_1 + \alpha_1^2 q_1^t\mathrm{M}q_1) \\ &= \frac{1}{\beta_1^2}(q_1^t\mathrm{M}^3 q_1 - 2\alpha_1\beta_1^2 + \alpha_1^3) \end{aligned} \tag{3.9}$$

Equation (3.9) follows from Equation (3.8), which can be written as $q_1^t\mathrm{M}^2 q_1 =$

$\alpha_1^2 + \beta_1^2$. The preceding paragraphs define all but one of the quantities in the last expression of (3.9). The one remaining is $q_1^t \mathrm{M}^3 q_1$, which can also be expressed in terms of covariances of ϕ, and the analysis is almost identical to that for $q_1^t \mathrm{M}^2 q_1$:

$$\begin{aligned} q_1^t \mathrm{M}^3 q_1 &= \frac{(\phi - \mathbf{E}_\pi \phi)^t \Pi \mathrm{P}^3 (\phi - \mathbf{E}_\pi \phi)}{(\phi - \mathbf{E}_\pi \phi)^t \Pi (\phi - \mathbf{E}_\pi \phi)} \\ &= \frac{\mathbf{C}_\pi(\phi(n), \phi(n+3))}{\mathbf{Var}_\pi(\phi(n))}. \end{aligned}$$

To generalize the foregoing for higher order Lanczos coefficients requires a recurrence relation between (α_{j+1}, β_j) and (α_j, β_{j-1}). As we will see, such a relation involves higher order Lanczos vectors q_j—vectors we do not wish to compute. Recall that the only "vector" we can handle is our initial observable. Therefore, we must also establish a recurrence among the q_j's, thus making them available inductively from our observable vector.

To begin with the β_j's, recall that $\beta_j = \|\mathrm{M}q_j - \alpha_j q_j - \beta_{j-1} q_{j-1}\|_2$. Whence,

$$\begin{aligned} \beta_j^2 &= (\mathrm{M}q_j - \alpha_j q_j - \beta_{j-1} q_{j-1})^t (\mathrm{M}q_j - \alpha_j q_j - \beta_{j-1} q_{j-1}) \\ &= q_j^t \mathrm{M}^2 q_j - \alpha_j^2 - \beta_{j-1}^2. \end{aligned} \tag{3.10}$$

Some simple algebra and the definitions $\alpha_j = q_j^t \mathrm{M} q_j$ and $\beta_{j-1} = q_j^t \mathrm{M} q_{j-1}$ prove equality (3.10). The recurrence for α_{j+1} is equally straight forward, though the

final form is not as neat as the expression above. Indeed,

$$\begin{aligned}\alpha_{j+1} &= q_{j+1}^t \mathrm{M} q_{j+1} \\ &= \frac{1}{\beta_j^2}(\mathrm{M}q_j - \alpha_j q_j - \beta_{j-1}q_{j-1})^t \mathrm{M}(\mathrm{M}q_j - \alpha_j q_j - \beta_{j-1}q_{j-1}) \\ &= \frac{1}{\beta_j^2}(q_j^t \mathrm{M}^3 q_j - 2\alpha_j q_j^t \mathrm{M}^2 q_j - 2\beta_{j-1}^2 q_{j-1}^t \mathrm{M}^2 q_j + \alpha_1^3 + 2\alpha_j\beta_{j-1}^2 + \beta_{j-1}^2). \quad (3.11)\end{aligned}$$

From (3.10), $\alpha_j^2 + \beta_j^2 + \beta_{j-1}^2$, and upon plugging this into (3.11), we arrive at

$$\alpha_{j+1} = \frac{1}{\beta_j^2}(q_j^t \mathrm{M}^3 q_j - \alpha_j^3 - 2\alpha_j\beta_j^2 - 2\beta_{j-1}q_{j-1}^t \mathrm{M}^2 q_j + \beta_{j-1}^2). \quad (3.12)$$

The expressions above involve the higher order Lanczos vectors q_j. In the present context, these vectors will not be readily available. However, another recursion—this time relating forms involving q_{j+1} to forms involving q_j and q_{j-1}—will make these expressions accessible by induction on our initial observable. The general forms of interest are $q_{j+1}^t \mathrm{M}^k q_{j+1}$ and $q_{j+1}^t \mathrm{M}^k q_j$. Again, we begin with the simpler of the two:

$$q_{j+1}^t M^k q_j = \frac{1}{\beta_j}(\mathrm{M}q_j - \alpha_j q_j - \beta_{j-1}q_{j-1})^t \mathrm{M}^k q_j$$

$$\frac{1}{\beta_j}(q_j^t \mathrm{M}^{k+1} q_j - \alpha_j q_j^t \mathrm{M}^k q_j - \beta_{j-1} q_j^t \mathrm{M}^k).$$

The second is,

$$
\begin{aligned}
q_{j+1}^t M^k q_{j+1} &= \frac{1}{\beta_j^2}(\mathrm{M}q_j - \alpha_j q_j - \beta_{j-1}q_{j-1})^t \mathrm{M}^k (\mathrm{M}q_j - \alpha_j q_j - \beta_{j-1}q_{j-1}) \\
&= \frac{1}{\beta_j^2}(q_j^t \mathrm{M}^{k+2} q_j + \alpha_j^2 q_j^t \mathrm{M}^k q_j + \beta_{j-1}^2 q_{j-1}^t \mathrm{M}^k q_{j-1} - 2\alpha_j q_j^t \mathrm{M}^{k+1} q_j \\
&\quad + 2\alpha_j \beta_{j-1} q_j^t \mathrm{M}^k q_{j-1} - 2\beta_{j-1} q_j^t \mathrm{M}^{k+1} q_{j-1}). \qquad (3.13)
\end{aligned}
$$

The point of all this is that each pair of coefficients (α_{j+1}, β_j) can be expressed as a function of the covariance between $\phi(n)$ and $\phi(n+m)$, for $m = 0, \ldots, 2j+1$. Since the expressions are based on four recurrence relations, they become quite complicated as j gets large. However, defining these recursions in a computer program is trivial, as we demonstrate in Section 4.4 by applying the foregoing to a particular problem.

Part II

Application

Chapter 4

Convergence to Gibbs measure on the Ising lattice

4.1 Physical Motivation

Before considering the details of the Metropolis algorithm, it helps to understand the setting in which it was developed. To do so we consider a typical example, the *Ising model*. This is a model of a system consisting of n "sites," each site taking values in $\{0, 1\}$. It might help to imagine these sites as equally spaced points on a line or a circle, though we are not restricted to such cases. The state space S consists of all possible n-dimensional permutations of 0's and 1's. Thus, $S = \{0, 1\}^n$ and $|S| = 2^n$. We can think of each site as a node existing in one of two possible positions, say "on" or "off." Each state $i \in S$ is a unique permutation σ_i of of 0's and 1's. Let $\sigma_n(k)$ denote the position of the kth node when the system is in state i. An observable $\phi = \phi(\sigma_i)$ on this space is a function of the permutations σ_i, $i \in \{1, \ldots, 2^n\}$.

The *energy* of the system, when in state i, is defined by the Hamiltonian

$$H(\sigma_i) = -\sum_{\langle jk \rangle} \sigma_i(j)\sigma_i(k)$$

where the sum runs over all nearest neighbor pairs. In statistical mechanics one is often concerned with the *Gibbs measure* of state i, which is defined by

$$\pi(\sigma_i) = Z^{-1} \exp\{-\beta H(\sigma_i)\} \tag{4.1}$$

where Z^{-1} is a normalizing constant, sometimes called the *partition function*. For the Ising model, we put

$$Z = \sum_{i=1}^{2^n} \exp\{-\beta H(\sigma_i)\}$$

so that π is a probability measure. Now, letting k denote the so called *Boltzmann constant*, if a classical mechanical system is in thermal equilibrium with its surroundings with absolute temperature T, and is in state i with energy $H(\sigma_i)$, then the probability density in phase-space of the point representing state i is given by (4.1) with $\beta = (kT)^{-1}$. A fundamental result of ergodic theory implies that we can also interpret $\pi(\sigma_i)$ as the proportion of time the system spends in state i. If the system is observed at a random time, the expected value $\mathbf{E}_\pi \phi$ of any observable ϕ is thus

$$\mathbf{E}_\pi \phi = \sum_{i=1}^{2^n} \phi(\sigma_i)\pi(\sigma_i) = Z^{-1} \sum_{i=1}^{2^n} \phi(\sigma_i) \exp\{-\beta H(\sigma_i)\}. \tag{4.2}$$

Perhaps the number of sites n in our Ising model is so large that it is impractical or impossible to evaluate (4.2). We might instead consider importance sampling and generate states with the probability density π given in (4.1). Then ϕ is itself an unbiased estimator of (4.2).

If evaluation of (4.2) is difficult, there is no reason to believe that evaluation of (4.1) is any easier. However, Metropolis and his collaborators [8] contrived a method for producing an ergodic Markov chain with transition probability matrix K whose elements $k(x, y)$ satisfy

$$\sum_x \pi(x)k(x, y) = \pi(x)$$

By Definition 1.4 this means that the Markov chain has stationary distribution π. Since the chain is ergodic, we know that it will eventually converge to the desired distribution. So, our main problem—the rate at which such a Markov chain will converge—is of primary concern in applications of the Metropolis algorithm. Moreover, as we will see below, the Metropolis algorithm ensures that the Markov chain has the desired stationary distribution by requiring that the chain satisfy the stronger condition of reversibility, or "detailed balance." Therefore, Markov chains produced by the Metropolis algorithm satisfy the assumptions of this paper, and our theory and methods can be used to bound convergence rates of such chains.

4.2 Description of the Metropolis Algorithm

The following explains how the Metropolis Algorithm is carried out. The exposition is similar to that given in Hammersley and Handscomb's book *Monte Carlo Methods* [7]. We refer the reader desiring more details to [7, Section 9.3].

To begin, choose an arbitrary symmetric Markov chain; that is, a Markov chain whose transition probability matrix P is symmetric. The matrix P is called the

proposition matrix, and its elements satisfy, for all i and j in S,

$$p(i,j) \geq 0, \quad p(i,j) = p(j,i), \quad \sum_m p(i,m) = 1. \tag{4.3}$$

Using P, we will derive a new transition probability matrix K with elements $k(i,j)$ satisfying

$$\sum_i \pi(i)k(i,j) = \pi(j) \tag{4.4}$$

For $i \neq j$ define

$$k(i,j) = \begin{cases} p(i,j)\pi(j)/\pi(i), & \text{if } \pi(j)/\pi(i) < 1, \\ p(i,j), & \text{if } \pi(j)/\pi(i) \geq 1. \end{cases} \tag{4.5}$$

For $i = j$ define

$$k(i,j) = p(i,i) + \sum_{j \in J_i} \big(1 - \frac{\pi(j)}{\pi(i)}\big) p(i,j),$$

where $J_i = \{j : j \neq i, \pi(j)/\pi(i) < 1\}$. Notice that $\pi(i) > 0$ implies $k(i,j) \geq 0$, and

$$\begin{aligned} \sum_j k(i,j) &= p(i,i) + \sum_{j \in J_i} \left\{ \big(1 - \frac{\pi(j)}{\pi(i)}\big) p(i,j) + \frac{p(i,j)\pi(j)}{\pi(i)} \right\} + \sum_{j \in J_i^c} p(i,j) \\ &= p(i,i) + \sum_{j \in J_i} p(i,j) + \sum_{j \in J_i^c} p(i,j). \end{aligned} \tag{4.6}$$

Equation (4.6) shows that for each i we have $\sum_j k(i,j) = 1$ (the row sums of K are 1), which confirms that K is a stochastic matrix.

Next we check that K satisfies the following *detailed balance* condition:

$$\pi(i)k(i,j) = \pi(j)k(j,i). \tag{4.7}$$

Obviously this holds when $i = j$, so assume $i \neq j$.

First suppose $\pi(i) = \pi(j)$. In this case, the definition (4.5) and symmetry of P imply that

$$k(i, j) = p(i, j) = p(j, i) = k(j, i).$$

Therefore, when $\pi(i) = \pi(j)$ the detailed balance condition (4.7) clearly holds.

Now suppose that $\pi(i) > \pi(j)$. Then the second case in definition (4.5) implies $k(j, i) = p(j, i)$, while the first case gives

$$k(i, j) = p(i, j)\pi(j)/\pi(i) = p(j, i)\pi(j)/\pi(i) = k(j, i)\pi(j)/\pi(i),$$

and we see that $\pi(i)k(i, j) = \pi(j)k(j, i)$ holds. Finally, a symmetric argument shows that if $\pi(j) > \pi(i)$, then detailed balance is again satisfied.

To complete the justification of the Metropolis algorithm, we note detailed balance (4.7) implies that π is indeed the stationary distribution for a Markov chain with transition probability matrix K; that is, (4.4) is satisfied. Indeed, by the detailed balance condition and the fact that the row sums of K are 1, we have

$$\sum_i \pi(i)k(i, j) = \sum_i \pi(j)k(j, i) = \pi(j) \sum_i k(j, i) = \pi(j).$$

We now summarize the tasks performed when implementing the Metropolis algorithm. First, pick as a proposition matrix any symmetric stochastic matrix and denote it by P. Begin the simulation of the Markov chain in state i, and then take one step of the proposition Markov chain to arrive in state j. That is, choose state j according to the probability mass function $p(i, \cdot)$. Next, compute $\pi(j)/\pi(i)$. If $\pi(j)/\pi(i) \geq 1$, accept j as the new state. If $\pi(j)/\pi(i) < 1$, then with probability

$\pi(j)/\pi(i)$ accept j as the new state; otherwise (with probability $1 - \pi(j)/\pi(i)$), take i as the new state. Performing each of these tasks produces one step of the Markov chain represented by the transition probability matrix K.

4.3 The Ising Model

Returning to the Ising model considered in Section 4.1, we consider how the Metropolis algorithm is applied to such a model. Again, start the simulated chain in state $X_0 = i$ and let $X_1 = j$ be a state chosen according to $p(i, \cdot)$, where $p(i, j) = p(j, i)$. Next, compute

$$\pi(\sigma_j)/\pi(\sigma_i) = \exp\{-\beta[H(\sigma_j) - H(\sigma_i)]\}$$

Suppose the energy $H(\sigma_j)$ of the new state satisfies $H(\sigma_j) \leq H(\sigma_i)$. Then $\pi(\sigma_j)/\pi(\sigma_i)$ is at least 1 and we move to state j. On the other hand, if $H(\sigma_j) > H(\sigma_i)$, then $\pi(\sigma_j)/\pi(\sigma_i) < 1$ and we take

$$X_1 = \begin{cases} j & \text{with probability } e^{-\beta\Delta H}, \\ i & \text{with probability } 1 - e^{-\beta\Delta H}. \end{cases} \tag{4.8}$$

where $\Delta H = H(\sigma_j) - H(\sigma_i)$ is the change in energy from state j to state i.

There are many ways to choose a proposition matrix P. The simplest derives from the so called *Glauber dynamics*, which proceeds as follows: choose one of the n sites at random with probability $1/n$—let s denote the chosen site—and change the position of this site; i.e., subtract its value from 1. The resulting state j is the

binary string given by

$$\sigma_j(k) = \begin{cases} 1 - \sigma_i(k) & \text{if } k = s, \\ \sigma_i(k) & \text{if } k = s, \end{cases} \qquad (k = 1, \ldots, d).$$

The procedure we employ below involves a slight variation: choose site s at random uniformly from the n possible sites. However, instead of changing the value of the chosen site with absolute certainty, as in Glauber dynamics, we "flip the switch" with probability 1/2. Therefore, when $k \neq s$ we have $\sigma_j(k) = \sigma_i(k)$, and when $k = s$,

$$\sigma_j(s) = \begin{cases} 1 - \sigma_i(s) & \text{with probability } 1/2, \\ \sigma_i(s) & \text{with probability } 1/2. \end{cases}$$

The advantage of this procedure is that the proposition matrix, before altering it with the Metropolis algorithm, is a bit more interesting than that produced by traditional Glauber dynamics. For, it has 1/2 along the main diagonal and is thus aperiodic and converges to a stationary distribution. Glauber dynamics has period 2, so its proposition matrix does not satisfy the hypotheses of our convergence theorem (Theorem 1.7). Keep in mind, however, that any proposition matrix modified by the Metropolis algorithm satisfies our hypotheses.

4.4 Computations

4.4.1 Computer Programs

Writing a computer program to simulate realizations of the Markov chain described above (i.e. with transition matrix K) is straight forward, and we do so using Matlab

and the program `ising.m` which appears in the Appendix.

When performing simulations in the present context, there are a few important aspects to consider. First, our program begins with a "hot start," which means that the initial state is picked by assigning each site to the value 0 or 1, each with probability 1/2.[1]

The method we have developed depends solely on covariances of our observable, which we estimate from the data produced by the simulations. However, such estimates require that the data come from the stationary distribution. Therefore, we must discard a number of observations before using the data to estimate covariances. The number of observations to discard depends on how long must we wait before we can expect the data to represent samples from the stationary distribution. When employing our procedure, we can discard a conservative (very large) number of observations and perform our analysis, deriving a bound on the convergence time. Then, in all future studies, we will know how much data should be discarded before samples can be assumed to have come from the stationary distribution.

In considering what observable to use on the state space described above, perhaps the most obvious is simply the number of nodes in the "on" position (i.e., the number of 1's). Recalling that $\sigma_j(k) \in \{0, 1\}$ denotes the position of the kth node when the system is in state j, this observable is simply the sum $\phi(j) = \sum_k \sigma_j(k)$. The program `ising.m` simulates the Markov chain, computes the values of this observable (and its square) and writes these data to a file called `phi.dat`. The program then computes the first few Lanczos coefficients using covariances pro-

[1] We cannot simply draw our initial state from the stationary distribution since the point of this work is to deal with cases for which samples from the stationary distribution are not immediately attainable. Furthermore, we are trying to determine how long it will take for observations from the simulated chain to represent samples from the stationary distribution.

Table 4.1: Estimates of $\lambda_{\max}(\mathrm{K}) = 0.998746$ using ϕ_1.

simulation #	α_1	β_1	$\lambda_{\max}(\mathrm{T}_2)$	β_2
1	0.995107	0.006960	0.995220	0.600243
2	0.995099	0.005891	0.994863	0.477393
3	0.994862	0.009671	0.994925	N/A
4	0.995321	0.009319	0.995654	N/A
5	0.994787	0.004872	0.405446	0.994742

duced by Matlab, so that we can check them against the results we get from the `lanczos.c` program, which we now describe.

The lanczos.c program (listed, along with its dependencies, in the Appendix) implements the recursive relations described in Section 3.2 to compute the Lanczos coefficients by the new method. It accepts input from the user specifying any observable, any number of iterations, and any number of desired Lanczos coefficients. That is, the observable could have come from a simulation of any reversible Markov chain and not just our special Metropolis chain. The output of this program is a matrix T containing the maximum number of Lanczos coefficients (up to the number requested by the user) before an approximate invariant subspace was reached. The eigenvalues of the nonzero part of this matrix, in accordance with the theory developed above, should provide a close approximation to the eigenvalues of the Markov chain's stochastic matrix.

4.4.2 Results

Table 4.4.2 shows the results of 5 simulations each of 100,000 iterations (discarding the first 5,000 observations) for the one dimensional Ising lattice with 10 nodes and $\beta = 1$. Displayed is the estimate of the first Lanczos coefficient α_1 (which is the eigenvalue estimate after one step; i.e., $\lambda_{\max}(\mathrm{T}_1)$), along with its error bound β_1.

Table 4.2: Estimates of $\lambda_{\max}(K) = 0.998746$ using ϕ_2

simulation #	α_1	β	$\lambda_{\max}(\mathrm{T}_2)$	β_2
1	0.994262	0.006142	0.994222	N/A
2	0.994333	0.002974	0.994332	N/A
3	0.994307	0.008892	0.994374	0.726320
4	0.994436	0.012884	0.994685	0.361404
5	0.993789	0.006909	0.993820	0.788710

Appearing in the third column is the largest eigenvalue, $\lambda_0(T_2)$, of the 2×2 matrix T_2, and β_2 appears in the fourth column when it is was computable. When it was not computable, it was because an approximate invariant subspace was reached at the first step, so β_2 involved expressions which were close to machine epsilon and thus could not be computed accurately. Table 4.4.2 shows the same information when using a second observable, $\phi_2 = \phi_1^2$ (i.e., the number of 1's squared).

The true value of the subdominant eigenvalue for this problem is $\lambda_{\max}(\mathrm{K}) = 0.998746$, and the next largest is $\lambda_2(\mathrm{K}) = 0.993303$. These were computed directly from the transition matrix using the Matlab program `tpm.m` listed in the Appendix. They should be very accurate, and it takes the Matlab routine `eig()` just over 20 minutes of cpu time on a Sun Ultra Sparc to find all the eigenvalues of K.

As mentioned in Section 3.1 (see Theorem 3.1), the β coefficients provide conservative error bounds on the eigenvalue estimates from the previous Lanczos step. Unfortunately, for this example β appears to be too conservative as it does not even allow us to bound our estimates away from 1. Furthermore, it seems that the estimates generated by the method above fall systematically below the true value of the subdominant eigenvalue. In every simulation, an approximate invariant subspace (to machine tolerance) was reached after computing at most 2 pairs of Lanczos coefficients. For ϕ_1 we see that it was reached after computing only

one pair on simulations 3 and 4, similarly for ϕ_2 on simulations 1 and 2. These observations indicate that the chosen observable is close to the slowest mode (or eigenfunction) of the process. That is, we should observe fast convergence to an approximate invariant space. However, the space found might contain only the second slowest mode of the process and, in that case, our eigenvalue estimates would be closer to the third largest eigenvalue, instead of the second largest.[2]

The matrix T was produced by the program `lanczos.c` and its eigenvalues were computed by Matlab, both operations taking a few seconds. Granted, comparing a few seconds to the 20 minutes it takes Matlab to compute all the eigenvalues is not really fair, since calling Matlab's `eig()` routine is not always the optimal traditional method for computing only a few eigenvalues of K. However, recall that the primary motivation for the new method are those examples where the matrix K is so large that we can't store even a single column vector in fast memory. Once we start swapping data to and from slow memory, traditional algorithms can become intractable, and such examples are easy to come by. For instance, if the Ising model described has 1000 nodes, it produces a $2^{1000} \times 2^{1000}$ transition probability matrix.

[2]Recall from above that we won't converge to the eigenspace corresponding to the largest eigenvalue (which is always 1), unless we choose a nearly constant observable function.

Chapter 5

Conclusion

We conclude with some remarks about the special situations in which our method is useful. We are interested in the subdominant eigenvalue of a large transition probability matrix. Our method can be used to approximate these eigenvalues when the following conditions are satisfied:

- The Markov chain is reversible (satisfies detailed balance).
- We can simulate realizations of the chain as well as functions of these realizations (observables on the state space).

When these conditions hold, we can avoid the traditional Lanczos algorithm, and estimate the values of the Lanczos coefficients via the variational properties of observables on the state space. There are a number of ideas in this paper that could be expanded upon in future studies. Perhaps most obvious would be to try to find analogous techniques for nonreversible (i.e. nonsymmetric) Markov chains. This is a considerable problem since the Lanczos methods described above are no longer applicable. Perhaps the *Arnoldi algorithm* (see [1, Algorithm 6.9]),

or the *nonsymmetric Lanczos algorithm* could also be modified to exploit special properties of stochastic matrices.

When applying the Lanczos algorithm, we usually choose a single starting vector (in the present context, an observable), perform the algorithm for a number of steps, and gather results. Then we do the same with a new starting vector and compare the results to those obtained with the first vector. Proceeding in this way for a number of starting vectors, we can provide statistical evidence that we have, indeed, located the eigenvalues of interest. However, when there are two observables, ϕ_1 and ϕ_2, of particular interest, perhaps we would benefit from considering both observables simultaneously. Future studies might develop an algorithm which incorporates both observables, again via $\psi_i = \Pi^{1/2}(\phi_i - \mathbf{E}_\pi \phi_i)$. The Lanczos algorithm would then be based on the space

$$\mathcal{K}^*(M, \psi_1, \psi_2, p) = \text{ran}\{\psi_1, \psi_2, \mathrm{M}\psi_1, \mathrm{M}\psi_2, \ldots, \mathrm{M}^{p-1}\psi_1, \mathrm{M}^{p-1}\psi_2\}$$

The Lanczos coefficients would then involve not only the autocovariances, but also the cross-covariances, $\mathbf{C}_\pi(\phi_i(n), \phi_j(n+m))$, defined in (2.6) above.

Appendix: computer programs

```
% Matlab code ising.m
%
% Written by William J. DeMeo on 12/15/97
% last modified 2013.10.19
% Inputs:
%         d = number of nodes of the ising lattice (e.g. d=100)
%         n = number of iterations (e.g. n=1000)
%         beta = Annealing schedule (e.g. beta = 2,
%            when beta -> 0 will always go to nee state
%            when beta -> infty will never go to state of higher energy)
%
X = zeros(d,1);
Energy = zeros(n,1);
Ave=0;

% Initialize using hot start
for i=1:d,
  if rand < .5
    X(i) = -1;
  else
    X(i) = 1;
  end
end

% Initialize at state of high energy (not used)
if 0
  X(1:2:d-1)=1;
  X(2:2:d)=1;
end

% Initialize energy
H=0;
for i=1:(d-1)
  H = H - X(i)*X(i+1);
end

% Initialize observable
phi = zeros(n+1,1);

for i=1:d
  if X(i) == 1
    phi(1)= phi(1)+1;
```

```
  end
end

for j=1:n
  site=0;
  while site==0
    site = round(rand*d);
  end
  % Compute new energy (with new X(site) = -X(site)):
  if site == 1
    Hnew = H + 2*X(1)*X(2);
  elseif site == d
    Hnew = H + 2*X(d)*X(d-1);
  else
    Hnew = H + 2*X(site)*(X(site-1) + X(site+1));
  end

  % Change to new state in two ways:
  % with probability 1/2, flip the switch
  if rand < .5
    if (Hnew <= H) | (rand < exp(-beta*(Hnew-H)))
      X(site) = -X(site);
      H = Hnew;
      % Compute new value of observable
      if X(site) == 1
        phi(j+1) = phi(j) + 1;
      else
        phi(j+1) = phi(j) -1;
      end
    else phi(j+1)=phi(j);
    end
  else phi(j+1)=phi(j);
  end
  Energy(j) = H;
end
plot(Energy)

% A second observable
phi2 = zeros(n+1,1);
for i=1:n+1
  phi2(i) = phi(i)*phi(i);
end

% Write first observable to phi.dat:
```

```
fid = fopen('phi.dat','w');
fprintf(fid,'%f\n',phi);
fclose(fid);

% Write second observable to phi2.dat:
fid = fopen('phi2.dat','w');
fprintf(fid,'%f\n',phi2);
fc1ose(fid);

% Compute a few Lanczos coefficients by new method
START=1001;
V1=zeros(2);
C1=zeros(2);
C2=zeros(2);
C3=zeros(2);

V1 = cov(phi(START:n+1),phi(START:n+1));
V1 = V1(1,1);
C1 = cov(phi(START:n),phi(START+1:n+1));
C1 = C1(1,2);
C2 = cov(phi(START:n-1),phi(START+2:n+1));
C2 = C2(1,2);
C3 = cov(phi(START:n-2),phi(START+3:n+1));
C3 = C3(1,2);

alpha1 = C1/V1;
beta1 = sqrt(C2/V1 -(C1/V1)^2);
alpha2 = (1/(beta1^2))*(C3/V1 - 2*alpha1*(C2/V1) + alpha1^3);

T1 = [alphal beta1; beta1 alpha2]
V1 = cov(phi2(START:n+1),phi2(START:n+1));
V1 = V1(1,1);
C1 = cov(phi2(START:n),phi2(START+1:n+1));
C1 = C1(1,2);
C2 = cov(phi2(START:n-1),phi2(START+2:n+1));
C2 = C2(1,2);
C3 = cov(phi2(START:n-2),phi2(START+3:n+1));
C3 = C3(1,2);

alpha1 = C1/V1;
beta1 = sqrt(C2/V1 - (C1/V1)^2);
alpha2 = (1/(beta1^2))#(C3/V1 - 2*alpha1*(C2/V1) + alpha1^3);

T2 = [alpha1 beta1; beta1 alpha2]
```

```
fid = fopen('Tmats.dat','v');
fprintf(fid,'%f\n',T1);
fprintf(fid,'%f\n',T2);
fclose(fid);

%%% end ising.m
```

```
% Matlab code tpm.m for generating transition probability matrix
% and computing its eigenvalues directly
%
% Written by William J. DeMeo on 1/10/98
%
% Inputs:
%        d = number of nodes of the ising lattice (e.g. d=10)
%     beta = Annealing schedule (e.g. beta - 2,
%            when beta -> 0 will always go to new state
%            when beta -> infty will never go to state of higher energy)

disp('bui1ding proposition matrix...')

states = 2^d;
A = 0;
for i=0:(d-1)
  E=eye(2^i);
  A = [A E; E A];
end

B = .5*eye(states);
A = B + .5*(1/d)*A;
% our modified Glauber dynamics requires the B and the .5*(1/d)

disp('...done')

% display pattern of nonzero entries
% spy(A)

% check that all row sums are 1
check=0;
```

```
for i=1:states
  check = check + (not(sum(A(i,1:states))<.99));
end
% if not all rows sum to 1, print check = (# of rows with sum=1)
if not(check==states)
  check
  error('Row sums are not all 1')
end

% construct matrix of states
E = zeros(d,states);
flip=-1;
for i=1:d
  for j=1:2^(i-1):states
    flip = -1*flip;
    for k=0:2^(i-1)
      E(i,j+k) = flip;
      end
  end
end
E=E';

% display first 64 states
% E(1:64,:)

% compute energy of each state
H = zeros(states,1);
for i=1:states
  for j=1:d-1
    H(i) = H(i) - E(i,j)*E(i,j+1);
  end
end

disp('building Hetropolis transition matrix...')

for i=1:states
  for j=i+1:states
    if not(A(i,j)==0)
      if(H(j)>H(i))
        alt = A(i,j)*(1-exp(-beta*(H(j)-H(i))));
        A(i,i) = A(i,i)+a1t;
        A(i,j) = A(i,j)-alt;
      elseif(H(j)<H(i))
        alt = A(j,i)*(1-exp(-beta*(H(i)-H(j))));
```

```
        A(j,j) = A(j,j)+a1t:
        A(j,i) = A(j,i)-alt;
      end
    end
  end
end

disp('...done')

check=0;
for i=1:states
  check = check + (not(sum(A(i,1:states))<.99));
end

% if not all rows sum to 1, print check = (# of rows with aum=1)

if not(check==states)
  check
  error('Row sums are not all 1')
end

disp('computing eigenvalues of tpm...')
cput = cputime;
evals = eig(A);
ecput = tputime - cput;
disp('the CPU time (in secs) for computing eigenvalues of tpm: ')
ecput

%%% end tpm.m
```

```
/*************************************************************
 * lanczos.c main program for computing Lanczos coefficients *
 *                                                           *
 * Created by William J. DeMeo on 1/7/98                     *
 * Last modified 2013.10.19                                  *
 *************************************************************/
```

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "prototypes.h"

/* Machine constants on dino (Sun Ultra 1 at Courant) */
#define MACHEPS 1.15828708535533604981e-16
#define SQRTEPS 1.07623746699106147048e-08
#define MAX_NAME 100

void read_name(char *);

/* Prototypes from the file functions.c */
double alpha(int j);
double betasq(int j);
double form1(int j, int k);
double form2(int j, int k);
double moment(double *data, long n,
              double *ave, double *var, double *cov, long k);

/* external variables to be used by functions */
double *phi,*cov, *var, *ave;

main()
{

  char *filename;
  FILE *ofp;
  double temp=0, *T;
  long i, j, nrow, nlanc, START;
  int flag=0;

  filename = cmalloc(MAX_NAME);

  printf("\nName of file containing observed values: ");
  read_name(filename);
  printf("\nTota1 number of observations in file (iterations): ");
  scanf("%u",&nrow);
  printf("\nNumber of leading observations to discard: ");
  scanf("%u",&START);
  printf("\nNumber of Lanczos coefficients desired: ");
  scanf("%u",&nlanc);
  while(2*nlanc > nrow-START)
```

```
    {
      printf("\nNot enough data for that many coefficients.\n");
      printf("\nEnter a smaller number of Lanczos coefficients: ");
      scanf("%u",&nlanc);
    }

  phi = dmalloc(nrow);
  cov=dmalloc(2*nlanc);
  var=dmalloc(1);
  ave=dmalloc(1);
  T = dmalloc(nlanc*nlanc);
  for(i=0;i<nlanc;i++)
    T[i]=(double) 0;

  /* observable phi is stored contiguously column-wise by MATLAB */
  matlabread(phi, nrow, 1, filename);

  /* send the observable, offset by START */
  moment(phi+START,nrow-START,ave, var, cov,2*nlanc-1);

  /* First column of T */
  T[0] = alpha(1);
  if((temp=betasq(1))>MACHEPS*10)
    {
      T[1]=sqrt(temp);

      /* General column of T */
      j=2:
      for(i=1; i<nlanc-1, flag==0;)
{
  T[i*nlanc+i-1]=sqrt(betasq(i));
  T[i*nlanc+i] = alpha(i+1);
  if((temp = betasq(i+1))>MACHEPS*10)
    {
      T[i*nlanc+i+1] = sqrt(temp);
      i++;
    }
  else
    flag=1;
}

      /* Last column */
      if(flag!=1 && ((temp= betasq(nlanc-1))>MACHEPS*10))
{
```

```
  T[nlanc*nlanc-2] = sqrt(temp);
  T[nlanc*nlanc-1] = alpha(nlanc);
  printf("\nbeta(1) = %1f\nbeta(%d) = %lf (last beta)",
 T[1],(nlanc-1),T[nlanc*nlanc-2]);
}
      else
{
  printf("\nApproximate invariant space reached at step %d.",i);
  printf("\nbeta(1) = %lf\nbeta(%d) = %lf (last accurate beta)".
 T[1],i,T[i*nlanc+i-1]);
  printf("\nbeta(%d)^2: %lf (first spurious resu1t)",i+1,temp);
}
      printf("\nThe matrix T is: \n");
      matprint(T,nlanc,nlanc);
    }
  else
    {
      printf("\nmain(): Approximate invariant space reached at first step.");
      printf("\nalpha(1) - %lf\nbeta(1)^2: %lf (first spurious result)",
             T[0], temp);
    }
  ofp = fopen("Tmat.m","v");
  check(ofp);
  matlabwrite(T,nlanc,nlanc,ofp);
  fclose(ofp);
}

void read_name(char *name)
{
  int c, i = 0;
  while ((c = getchar()) != EOF && c != ' ' && c != '\n')
    name[i++] = c;
  name[i] = '\0';
}

/** end lanczos.c **/
```

```
/************************************************
 * functions.c -- functions required by lanczos.c  *
 *                                                 *
 * Created by William J. DeMeo on 1/7/98           *
 * Last modified 2013/10/19                        *
 ************************************************/

#include <math.h>
#define START 1000
#define ITER 10000

/* Machine constants on dino (Sun Ultra 1 at Courant) */
#define MACHEPS 1.158287085355336049 81e-16
#define SQRTEPS 1.076237466991061470 48e-08

double alpha(int j);
double betasq(int j);
double form1(int j, int k);
double form2(int j, int k);
double moment(double *data, long n,
              double *ave, double *var, double *cov, int k);

/* external variables to be used by functions */
double *phi,*cov, *var, *ave;

double alpha(int j)
{
  /* alpha is never called with j < 1 */
  if(j==1)
    return form1(1,1);
  else if(j>1)
    return
      ((double)1/betasq(j-1)) * (form1(j-1,3) - pow(alpha(j-1),3)
       - 2 * (alpha(j-1)*betasq(j-1) + sqrt(betasq(j-2))*form2(j-1,2))
       + betasq(j-2));
}


double betasq(int j)
{
  if(j==0)
    return (double)0;
  else if(j>0)
```

```
    return (form1(j,2) - pow(alpha(j),2) - betasq(j-1));
}


double form1(int j, int k)
{
  double form14, alpha1, alpha1sq, form12, form13;
  if(j==0)
    return (double)0;
  else if(j==1)
    {
      /* printf("\nvar = %lf, cov(%d) = %lf \n",*var,k,cov[k]); */
      return (cov[k])/(*var); /* the only real value */
    }
  else if(j>1)
    {
      return
((double)1/betasq(j-1))
* ( form1(j-1,k+2) + pow(alpha(j-1),2) * form1(j-1,k)
    + 2*(alpha(j-1) * sqrt(betasq(j-2)) * form2(j-1,k)
 - alpha(j-1)*form1(j-1,k+1)
 - sqrt(betasq(j2))*form2(j-1,k+1) )
    + betasq(j-2)*form1(j-2,k));
    }
}


double form2(int j, int k)
{
/* form2 is never called with j < 1 */
  if(j==1)
    return (double)0;
  else if(j>1)
    return
      (pov(betasq(j-1),-.5)) 3
      (form1(j-1,k+1) - alpha(j-1)*form1(j-1,k)
       - sqrt(betasq(j-2)) * form2(j1,k));
}

/* moment() function for computing var and cov(k)
   arguments:
   data = a nxi array of doubles
   n = length of data[]
   ave =(on exit)= the average of data[]
```

```
   var =(on exit)= the variance of data[]
   cov =(on exit)= the covariance of data[i] and data[i+j] for i-1,...,k
   k = the max lag for cov above
*/
double moment(double *data, long n,
              double *ave, double *var, double *cov, int k)
{
  /* Centered about data[0] algorithm: */
  long i,j;
  double ave1, ave2;

  *ave=0;*var=0; ave1=ave2=0;
  for(j=0;j<=k;j++)
    cov[j]=0;

  for(i=1;i<n;i++)
    {
      *ave += (data[i] - data[0]);
      *var += (data[i] - data[0])*(data[i] - data[0]);
    }
  *var /= (double)(n-1);
  *var -= (((*ave)/(double)n) * ((*ave)/(double)(n-1)));
  /* *ave = ((*ave)/(double)n) + data[0]; (the true average; not needed)*/

  for(j=0;j<=k;j++)
    {
      for(i=0;i<(n-j);i++)
{
  ave1 += (data[i] - data[0]);
  ave2 += (data[i+j] - data[0]);
  cov[j] += (data[i] - data[0])*(data[i+j] - data[0]);

}

      ave1/=(double)(n-j); ave2/=(double)(n-j);
      cov[j] = ((cov[j] - (double)(n-j)*ave1*ave2)/(double)(n-j-1));
    }

}

/*** end funccions.c ***/
```

Bibliography

[1] J. Demmel. *Applied Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, 1997.

[2] P. Diaconis and L. Saloff-Coste. Logarithmic Sobolev inequalities for finite Markov chains. *The Annals of Applied Probability*, 6:695–750, 1996.

[3] R. Durrett. *Probability: Theory and Examples*. Duxbury Press, second edition, 1996.

[4] S. Geman and D. Geman. Stochastic relaxation. Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[5] G. Golub and Charles Van Loan. *Matrix Computations*. Johns Hopkins University Press, third edition, 1996.

[6] G. Goodman and A. Sokal. Multigrid Monte Carlo method. conceptual foundations. *Physical Review D*, 40:2035–2071, 1989.

[7] J. M. Hammersley and D. C. Handscomb. *Monte Carlo Methods*. Chapman and Hall, 1964.

[8] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *J. Chem. Phys.*, 21:1087–1092, 1953.

[9] J. Rosenthal. Convergence rates for Markov chains. *Siam Review*, 37:387–405, 1995.