2007

The Lanczos algorithm and fermat's factoring method

Parvaneh Darafshi
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_theses

Recommended Citation

Darafshi, Parvaneh, "The Lanczos algorithm and fermat's factoring method" (2007). *Master's Theses*. 3585.

DOI: <https://doi.org/10.31979/etd.kqmh-m93t>
https://scholarworks.sjsu.edu/etd_theses/3585

This Thesis is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Theses by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

THE LANCZOS ALGORITHM AND FERMAT'S FACTORING METHOD

A Thesis

Presented to

The Faculty of the Department of Mathematics

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Parvaneh Darafshi

August 2007

UMI Number: 1448855

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI[®]

UMI Microform 1448855

Copyright 2007 by ProQuest Information and Learning Company.

All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

© 2007

Parvaneh Darafshi

ALL RIGHTS RESERVED

APPROVED FOR THE DEPARTMENT OF MATHEMATICS

A handwritten signature in black ink, appearing to read 'Timothy Hsu', written over a horizontal line.

Dr. Timothy Hsu

A handwritten signature in black ink, appearing to read 'Jane Day', written over a horizontal line.

Dr. Jane Day

A handwritten signature in black ink, appearing to read 'Les Foster', written over a horizontal line.

Dr. Les Foster

APPROVED FOR THE UNIVERSITY

A handwritten signature in black ink, appearing to read 'Paula C Stark', written over a horizontal line.

ABSTRACT

THE LANCZOS ALGORITHM AND FERMAT'S FACTORING METHOD

by Parvaneh Darafshi

In linear algebra there are different methods for solving systems of linear equations. Currently mathematicians are attempting to invent more accurate and less memory consuming algorithms that can be executed by computers, especially for the systems with large and sparse coefficient matrices. Mathematicians have found the algorithms based on iterative methods more convenient and faster than the straight forward ones, such as Gaussian elimination method. One of the preferred algorithms among the iterative methods is the Lanczos Algorithm.

In this thesis we present a *generalized* version of the Lanczos Algorithm. We define a successful run of this algorithm and then will prove to the definition, that is The Lanczos algorithm succeeds if and only if the vector at the algorithm's stop point is zero. We are calling our version *generalized* since the matrix involved in the algorithm does not have to be *positive definite* which is not definable over every arbitrary field. In the *generalized* version of the Lanczos algorithm we define a symmetric bilinear form over any arbitrary vector space, where the involved matrix A has to be self-adjoint relative to the defined bilinear form.

Finally, we discuss applications of the Lanczos algorithm. Specifically, the application of the Lanczos algorithm over finite fields towards factoring large integers is discussed to explain why factoring large integers can be a threat for computers' security, so it is important to know how difficult it would be for the intruders to do the factoring of the computer encoding password number.

DEDICATION

“Be the change you wish to see in The World” Mahatma Gandhi (1869-1948)

To all the people who are working hard by bringing positive changes to our
lives.

ACKNOWLEDGEMENTS

Thank you to everyone who helped me to achieve my youth days withheld degree, especially my dear family, Ehsan, Azar, Roozbeh and Pirooz.

TABLE OF CONTENTS

CHAPTER	
1	LINEAR ALGEBRA 1
1.1	Bilinear Forms 1
1.2	Isotropic Vectors, Orthogonality, and Non-Degeneracy 4
1.3	Krylov Subspaces 5
2	LANCZOS 8
2.1	Introduction 8
2.2	The Generalized Lanczos Algorithm 10
2.3	Using the Lanczos Algorithm to Solve $Ax = b$ 16
2.4	Tridiagonalization by the Lanczos Algorithm 20
2.5	Tridiagonalization by Normalized Lanczos Vectors 22
2.6	Normalized Lanczos Algorithm 23
3	NUMBER THEORY 26
3.1	Introduction 26
3.2	RSA Cryptography 27
3.3	Factoring 29
3.4	Quadratic Residues and Fermat's Square Pair Factorization 30
3.5	Quadratic Sieve 33

4	FERMAT MEETS LANCZOS	39
4.1	Review of Algebraic Problems Required for Quadratic Sieve	39
4.2	Reducing $Ax = 0$ to $Ax = b \neq 0$	40
4.3	Reducing $Ax = b$ for Non-Symmetric A to $Bx = c$ for Symmetric B .	41
4.4	Avoiding Self-Conjugacy	42
	 BIBLIOGRAPHY	 44
	 APPENDIX	
A	BACKGROUND AND NOTATION	46

CHAPTER 1

LINEAR ALGEBRA

We review standard background material from linear algebra. The main reference books are: Artin [Art91] and Friedberg/Insel/Spence [FIS97].

1.1 Bilinear Forms

In linear algebra, the notions of length, distance, and orthogonality in a vector space are introduced by using the ideas of *inner product* and *bilinear form*. This section follows Artin [Art91] Ch. 7 and Friedberg/Insel/Spence [FIS97] Ch. 6.7.

Definition 1.1.1. [FIS97] p. 381. Let V be a vector space over the field F . A function $H : V \times V \rightarrow F$ is called a *bilinear form* on V if H is linear in each variable when the other one is held fixed; that is,

$$(1) \quad H(ax_1 + x_2, y) = aH(x_1, y) + H(x_2, y) \text{ for all } x_1, x_2, y \in V \text{ and } a \in F, \text{ and}$$

$$(2) \quad H(x, ay_1 + y_2) = aH(x, y_1) + H(x, y_2) \text{ for all } x, y_1, y_2 \in V \text{ and } a \in F.$$

Example 1.1.2. The familiar standard dot product $H(x, y) = x^T y$ defined over the field F is a bilinear form on the vector space $V = F^n$.

Definition 1.1.3. [FIS97] p. 386. A bilinear form H over a vector space V is *symmetric* if $H(x, y) = H(y, x)$ for all $x, y \in V$.

Definition 1.1.4. Analogously to the definition of matrix representation for a linear transformation, for a bilinear form H on a vector space V with an ordered basis $\beta = \{v_1, v_2, \dots, v_n\}$, the $n \times n$ matrix A with entry $a_{ij} = H(v_i, v_j)$ is called the *matrix representation of H relative to the basis β* .

Definition 1.1.5. An n -by- n matrix A is called *symmetric* if $A^T = A$.

Example 1.1.6. Let $F = Z_2$, the field with elements $\{0, 1\}$, let $V = F^2$, and let $H : V \times V \rightarrow F$ be defined as: $H((a_1, a_2), (b_1, b_2)) = a_1b_2 + a_2b_1$.

Example 1.1.7. Let $F = R$, $V = R^3$, and define $H : V \times V \rightarrow F$ as:

$$H((a_1, a_2, a_3), (b_1, b_2, b_3)) = 2a_1b_1 + a_1b_2 + a_2b_1 + a_2b_2 + a_3b_3. \quad (1.1)$$

Example 1.1.8. Let $V = F^n$. For any $n \times n$ symmetric matrix A , let define the function $H : V \times V \rightarrow F$ as: $H(x, y) = x^T Ay$.

It is easy to see that H is a *symmetric bilinear form* on V in all three examples.

Actually, the first two examples are special cases of the third one; for example 1.1.6,

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \text{ and for example 1.1.7, } A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

But there is a hidden fact in example 1.1.7, which is not true about the example 1.1.6, and that is $H(x, x) \neq 0$ for all nonzero vectors $x \in V$. This property fails in the first example, since $H(x, x) = 0$ for $x = (1, 0)^T$.

Theorem 1.1.9. [FIS97] p. 387. *A bilinear form H over a vector space V is symmetric if and only if M , the matrix representation of H relative to any ordered basis $\beta = \{v_1, v_2, \dots, v_n\}$, is symmetric.*

Proof. Let the bilinear form H over vector space V be symmetric. Then by definition of M we will have $m_{ij} = H(v_i, v_j) = H(v_j, v_i) = m_{ji}$, implying that M is symmetric.

Conversely, if M is symmetric, then $H(v_i, v_j) = H(v_j, v_i)$ for all $1 \leq i, j \leq n$, which leads to the conclusion that $H(x, y) = H(y, x)$ for all x and y in V , since every vector in V is a linear combination of β , and H is bilinear. $\square$

Corollary 1.1.10. *Consider $H(u, v) = u^T A v$ over the vector space $V = F^n$, where A is an n -by- n matrix. Then H is symmetric if and only if A is symmetric.*

Proof. By definition 1.1.4, it is easy to see that the matrix A is the matrix representation of H relative to the standard basis, since for $u = e_i$ and $v = e_j$,

$$H(e_i, e_j) = e_i^T A e_j = a_{ij}. \quad (1.2)$$

So it follows from theorem 1.1.9 that A is symmetric. $\square$

Definition 1.1.11. Let V be a vector space over R . A symmetric bilinear form $H(x, y) : V \times V \rightarrow R$ is called an *inner product* on V if $H(x, y)$ is positive definite. This means that $H(x, x) > 0$ for all nonzero $x \in V$. Such a space is called an inner product space.

Example 1.1.7 is an inner product. In general the bilinear form $H(x, y)$ defined as $H(x, y) : R^n \times R^n \rightarrow R$ and $(x, y) = x^T y$ (the *standard dot product*) is an inner product; see Artin [Art91], Chapter 7.

Definition 1.1.12. An $n \times n$ real symmetric matrix A is *positive definite* if the bilinear form $H(x, y) = x^T A y$ on R^n is positive definite.

Example 1.1.13. It is easy to see that the $n \times n$ symmetric matrix I is positive definite over $V = R^n$, since the bilinear form $(x, y) = x^T A y$ with $A = I$ is a symmetric positive definite form on R^n . Also the matrix A in example 1.1.7 is positive definite, since for every $v = (a, b, c)^T \neq 0 \in R^3$, $H(v, v) = a^2 + (a + b)^2 + c^2 > 0$.

Definition 1.1.14. An n -by- n matrix A is called *self-adjoint relative to the symmetric bilinear form H* if $H(Ax, y) = H(x, Ay)$ for all $x, y \in V$.

Theorem 1.1.15. If A is an $n \times n$ symmetric matrix, and either $H(x, y) = (x, Ay)$ or $H(x, y) = (x, y)$, the well-known dot product, then A is self-adjoint relative to the bilinear form H .

Proof. If $H(x, y) = (x, Ay)$, then

$$H(Au, v) = (Au, Av) = u^T A^T Av = u^T A^2 v = H(u, Av), \quad (1.3)$$

since A is symmetric. Also if $H(x, y) = (x, y)$, then

$$H(Au, v) = (Au, v) = u^T A^T v = u^T Av = H(u, Av). \quad (1.4)$$

Therefore A is self-adjoint relative to H . □

Definition 1.1.16. For V a real vector space and $\langle u, v \rangle$ an inner product over V , the length of a vector $v \in V$ is defined to be

$$|v| = \sqrt{\langle v, v \rangle} = v^T v. \quad (1.5)$$

1.2 Isotropic Vectors, Orthogonality, and Non-Degeneracy

In this section, we mostly follow Artin [Art91].

Definition 1.2.1. For a bilinear form H over vector space V , the nonzero vector $x \in V$ is *isotropic*, or *self-conjugate*, if $H(x, x) = 0$; else x is *non-isotropic*. A set $\{v_1, v_2, \dots, v_n\}$ of vectors is non-isotropic if each v_i is non-isotropic.

Definition 1.2.2. A symmetric bilinear form H on a vector space V is *non-degenerate* if for every nonzero vector $x \in V$ there exists a vector $y \in V$ such that $H(x, y) \neq 0$. Otherwise H is *degenerate*.

Definition 1.2.3. [Art91] p. 241. In a vector space V the set $\{v_1, v_2, \dots, v_n\}$ is called *orthogonal* relative to the symmetric bilinear form $\langle u, v \rangle$ if for every $i \neq j$ we have $\langle v_i, v_j \rangle = 0$.

Theorem 1.2.4. *Every non-isotropic orthogonal set $\{v_1, v_2, \dots, v_n\}$ is linearly independent.*

Proof. Let $\alpha_1 v_1 + \alpha_2 v_2 + \dots + \alpha_i v_i + \dots + \alpha_n v_n = 0$ for some scalars α_i . By bilinear form properties and the orthogonality of the v_i , we can write:

$$\langle \alpha_1 v_1 + \alpha_2 v_2 + \dots + \alpha_i v_i + \dots + \alpha_n v_n, v_i \rangle = \alpha_i \langle v_i, v_i \rangle = 0, \quad (1.6)$$

implying that $\alpha_i = 0$, since $\langle v_i, v_i \rangle \neq 0$. Therefore, the set $\{v_1, v_2, \dots, v_n\}$ is linearly independent. $\square$

Theorem 1.2.5. *Consider a symmetric bilinear form $H(u, v) = \langle u, v \rangle$ defined on a vector space V . If there exists a non-isotropic orthogonal basis for V relative to H , then H is non-degenerate over V .*

Proof. Let the set $\{v_1, v_2, \dots, v_n\}$ be a non-isotropic orthogonal basis for V relative to H . Assume $\langle x, y \rangle = 0$ for all $y \in V$; then we have to show $x = 0$ to conclude the non-degeneracy of H . We can write $x = \alpha_1 v_1 + \alpha_2 v_2 + \dots + \alpha_i v_i + \dots + \alpha_n v_n$ since $x \in V$. Now by assumption $\langle x, v_i \rangle = 0$, so since the v_i are orthogonal, we have

$$\langle \alpha_1 v_1 + \alpha_2 v_2 + \dots + \alpha_i v_i + \dots + \alpha_n v_n, v_i \rangle = \alpha_i \langle v_i, v_i \rangle = 0. \quad (1.7)$$

Since v_i is non-isotropic, $\alpha_i = 0$ for all i . Therefore H is non-degenerate on V . $\square$

1.3 Krylov Subspaces

This section follows Golub/Van Loan [GL96] Chapters 9 and 10, and Saad [Saa03] Chapters 6 and 7.

Definition 1.3.1. [Saa03] p. 152. Let $b \in F^n$, and A be an $n \times n$ matrix with entries in F . The vector space $W = \text{Span}\{b, Ab, A^2b, A^3b, \dots\}$ is called the *Krylov subspace* generated by the vector b under the transformation matrix A . We will denote this subspace by $K(A, b)$.

Definition 1.3.2. [Saa03] p. 152. The m -Krylov subspace $K(A, b, m)$ is defined by $K(A, b, m) = \text{Span}\{b, Ab, A^2b, \dots, A^{m-1}b\}$.

It is easy to see that the Krylov subspace $K(A, b)$ is a finite-dimensional subspace of F^n . It is also easy to see that $K(A, b, m)$ is the column space of the n -by- m matrix $\kappa(A, b, m) = [b, Ab, \dots, A^{m-1}b]$; in other words, $K(A, b, m) = \text{Col}(\kappa(A, b, m))$. We will show in Theorem 1.3.4 that $K(A, b) = K(A, b, m)$ for some $m \leq n$, specifically, the smallest m such that $A^m b \in K(A, b, m)$.

Theorem 1.3.3. *The Krylov subspace $K(A, b)$ is the smallest A -invariant subspace containing b .*

Proof. If $x \in K(A, b)$, then x is of the form $x = \sum_{i=0}^k \alpha_i (A^i b)$, so

$$Ax = \sum_{i=0}^k \alpha_i (A^{i+1} b) \quad (1.8)$$

implies that $Ax \in K(A, b)$. Therefore, $K(A, b)$ is A -invariant. Conversely any A -invariant subspace containing b must contain Ab , $A(Ab) = A^2b$, A^3b , and so on, which means that $K(A, b)$ is the smallest A -invariant subspace containing b . $\square$

Any Krylov subspace generated by an $n \times n$ matrix A over an arbitrary field F has dimension $m \leq n$, since it is a subspace of the finite dimensional vector space F^n . The following theorem explains what m is.

Theorem 1.3.4. *For an n -by- n matrix A , and a nonzero vector b , let m be the largest integer such that the set $W_m = \{b, Ab, A^2b, \dots, A^{m-1}b\}$ is linearly independent. Then W_m is a basis for $K(A, b)$; in particular, $\dim K(A, b) = m$.*

Proof. Since F^n is a finite dimensional vector space, we know there exists a largest positive integer $m \leq n$ such that $W_m = \{b, Ab, A^2b, \dots, A^{m-1}b\}$ is linearly independent.

Let $S = \text{Span } W_m$. Since W_m is a basis for S , by the Dependency Theorem A.0.8, $W_m \cup \{A^m b\}$ is linearly dependent if and only if $A^m b \in S$. By construction, m is the largest integer such that W_m is linearly independent, so $W_{m+1} = W_m \cup \{A^m b\}$ is linearly dependent, which implies that $A^m b \in S = \text{Span } W_m$, and we can write

$$A^m b = \alpha_0 b + \alpha_1 (Ab) + \dots + \alpha_{m-1} (A^{m-1}b) \quad (1.9)$$

for some scalars α_i . Now consider any vector $x \in S$. We have that for some scalars β_i ,

$$Ax = A \sum_{i=0}^{m-1} \beta_i (A^i b) = \sum_{j=1}^{m-1} \beta_{j-1} (A^j b) + \beta_{m-1} A^m b. \quad (1.10)$$

By substituting $A^m b$ from equation (1.9) we see that $Ax \in S$. It follows that S is an A -invariant subspace of F^n . Since $K(A, b)$ is the smallest A -invariant subspace containing b , then $K(A, b) \subseteq S$. Conversely, $S = K(A, b, m) \subseteq K(A, b)$. Therefore, $S = K(A, b)$, and W_m is a basis for $S = K(A, b)$. $\square$

CHAPTER 2

LANCZOS

2.1 Introduction

In linear algebra, one standard problem is to find a vector x such that $Ax = b$ for a given m -by- n matrix A and an n -by-1 column vector b . The method of Gaussian elimination is the most straightforward method, but applying this method for a large matrix requires a large amount of computer time and memory, so mathematicians often look for faster methods.

Here is where the Cayley-Hamilton theorem comes as a time and life saver; see Ipsen/Meyer [IM]. The Cayley-Hamilton theorem states: Every n -by- n matrix A satisfies its characteristic polynomial; that is, if

$$f(t) = \det(A - tI) = a_n t^n + a_{n-1} t^{n-1} + \dots + a_1 t + a_0 \quad (2.1)$$

is the characteristic polynomial of A , then $f(A) = 0$, the n -by- n zero matrix. In particular, if A is a real nonsingular n -by- n matrix, for some $a_i \in R$, we can write

$$a_n A^n + a_{n-1} A^{n-1} + \dots + a_1 A + a_0 = 0. \quad (2.2)$$

So, by multiplying both sides by A^{-1} , we have

$$A^{-1} = \beta_{n-1} A^{n-1} + \beta_{n-2} A^{n-2} + \dots + \beta_0, \quad (2.3)$$

where $\beta_i = -\frac{a_{i+1}}{a_0}$. (Note $a_0 = \det A \neq 0$.) Using this equation to calculate $x = A^{-1}b$ discloses the importance of Krylov subspaces; specifically, since

$$A^{-1}b = \beta_{n-1}A^{n-1}b + \beta_{n-2}A^{n-2}b + \dots + \beta_1Ab + \beta_0b, \quad (2.4)$$

it follows that $x = A^{-1}b \in K(A, b)$.

In Lanczos' original article [Lan50] where he devised his algorithm to solve the Fredholm Problem ($y - \lambda Ay = b$) using an iterative method now known as the Lanczos algorithm, he notices that y has to be a linear combination of iterative vectors $b, Ab, A^2b, \dots$. He solves this system of equations by trying to minimize the length of a new vector generated from a linear combination of all iterated vectors before the last one. In this process, he is also looking to generate new minimized vectors that are orthogonal to all of the previously generated ones.

As we will see, because the Lanczos method is based on orthogonality, it works over any arbitrary field F , unless the existence of isotropic vectors or the degeneracy of bilinear forms causes problems. To eliminate nondegeneracy in vector spaces over R , Lanczos uses positive definite matrices. However, even though no such thing as matrix A being positive definite is definable in a vector space over finite fields, still this algorithm is useful in factoring large composite numbers or calculating discrete logarithms; see Coppersmith/Odlyzko/Schroeppel [COS86] and La Macchia/Odlyzko [LO].

Lanczos gave his algorithm for vector spaces over R , where the iterative calculation in R is always threatened by round-off errors. This threat is inactive in finite fields, in which we are most interested, but there exist other threats. So as we shall see the Lanczos algorithm will not always produce an answer for a given linear system of equations; see Coppersmith/Odlyzko/Schroeppel [COS86].

2.2 The Generalized Lanczos Algorithm

Here, we present the well-known Lanczos algorithm given in other sources, over an arbitrary field, but with a different approach. This version is not presenting a new theorem, but we are giving detailed proofs for the algorithm in a general form.

Searching through the standard references for a better understanding of this algorithm, we noticed that while the fundamental material is well-known and very similar, it is hard to find a standard reference in our level of generality. Even if we can see this point of the view among some sources, it is not usually presented with a proof. Compare Saad [Saa03], Pissanetsky [Pis84], Coppersmith/Odlyzko/Schroeppel [COS86], Golub/Van Loan [GL96] and Briggs [Bri98]. We can say [Lan50] is the closest to our generalized version.

Algorithm 2.2.1. Generalized Lanczos algorithm

Input : Any symmetric bilinear form $\langle u, v \rangle$ on F^n , an n -by- n matrix A with entries in F that is self-adjoint relative to $\langle *, * \rangle$, and a nonzero n -by-1 column vector b .

Procedure :

Let $w_0 = b$. If $\langle w_0, w_0 \rangle \neq 0$, let

$$w_1 = Aw_0 - \frac{\langle Aw_0, w_0 \rangle}{\langle w_0, w_0 \rangle} w_0. \quad (2.5)$$

For $i > 0$, if $\langle w_i, w_i \rangle \neq 0$, calculate

$$w_{i+1} = Aw_i - \frac{\langle Aw_i, w_i \rangle}{\langle w_i, w_i \rangle} w_i - \frac{\langle Aw_i, w_{i-1} \rangle}{\langle w_{i-1}, w_{i-1} \rangle} w_{i-1}. \quad (2.6)$$

Stop :

This algorithm stops when for some $m \geq 0$, $\langle w_m, w_m \rangle = 0$.

Output :

The output is the set $W = \{w_0, w_1, \dots, w_{m-1}\}$.

Definition 2.2.2. If the Lanczos algorithm stops because $w_m = 0$, then it has *succeeded*. Otherwise, (i.e., if w_m is nonzero but isotropic), it is a *failure*. Note that in this algorithm, if the initial vector $w_0 \in F^n$ is isotropic, then the set W is the empty set. Therefore, for the rest of the chapter, a successful algorithm will have $m \geq 1$.

We will show that Lanczos succeeds if and only if W is a basis for $K(A, b)$.

Definition 2.2.3. We use $GF(q) = F_q$ to denote the field with q elements.

Example 2.2.4. Let $F_3 = Z/(3Z)$ be the field with 3 elements. Consider the dot product as a symmetric bilinear form in F_3^3 . Let $b = w_0 = (1, 1, 1)^T \in F_3^3$, and

consider the symmetric matrix $A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ with entries in F_3 . By straightforward

calculation we see that $\langle w_0, w_0 \rangle = 0$, implying that $W = \{\}$, whereas $K(A, b) = \text{Span}\{(1, 1, 1)^T, (2, 0, 1)^T, (2, 2, 1)^T\}$ has dimension 3. We see that if Lanczos algorithm fails, then the output set W_m may not be a basis for $K(A, b)$.

Theorem 2.2.5. *At each step of the Lanczos algorithm up to the stopping point ($1 \leq k \leq m$), the non-isotropic set $W_k = \{w_0, w_1, \dots, w_{k-1}\}$ is linearly independent and orthogonal relative to $\langle *, * \rangle$. Furthermore, $\langle w_i, w_m \rangle = 0$ for $0 \leq i < m$, i.e., w_m is orthogonal to every vector in W_m .*

Proof. First we show that W_k is orthogonal for $1 \leq k \leq m$, proceeding by induction on k . This holds vacuously for $k = 1$, so as the base case of our induction, we show $W_2 = \{w_0, w_1\}$ is orthogonal. For this, since

$$w_1 = Aw_0 - \frac{\langle Aw_0, w_0 \rangle}{\langle w_0, w_0 \rangle} w_0, \quad (2.7)$$

we have

$$\langle w_1, w_0 \rangle = \left\langle Aw_0 - \frac{\langle Aw_0, w_0 \rangle}{\langle w_0, w_0 \rangle} w_0, w_0 \right\rangle. \quad (2.8)$$

By bilinearity, we have

$$\langle w_1, w_0 \rangle = \langle Aw_0, w_0 \rangle - \frac{\langle Aw_0, w_0 \rangle}{\langle w_0, w_0 \rangle} \langle w_0, w_0 \rangle = 0, \quad (2.9)$$

since $\langle w_0, w_0 \rangle \neq 0$, or else the algorithm would have stopped at $m = 0$.

Now as our induction assumption, assume for some $1 < k \leq m$ that $\langle w_i, w_j \rangle = 0$ for all $i \neq j$, where $i, j < k$. We have to show that $\langle w_k, w_j \rangle = 0$ for all $j < k$. There are three cases for j : (1) $j < k - 2$, (2) $j = k - 2$, and (3) $j = k - 1$.

(1) Let $j < k - 2$. By the Lanczos recursive equation (2.6) we have

$$\begin{aligned} w_k = Aw_{k-1} - \frac{\langle Aw_{k-1}, w_{k-1} \rangle}{\langle w_{k-1}, w_{k-1} \rangle} w_{k-1} \\ - \frac{\langle Aw_{k-1}, w_{k-2} \rangle}{\langle w_{k-2}, w_{k-2} \rangle} w_{k-2}. \end{aligned} \quad (2.10)$$

So by bilinearity, we can write

$$\begin{aligned} \langle w_k, w_j \rangle = \langle Aw_{k-1}, w_j \rangle - \frac{\langle Aw_{k-1}, w_{k-1} \rangle}{\langle w_{k-1}, w_{k-1} \rangle} \langle w_{k-1}, w_j \rangle \\ - \frac{\langle Aw_{k-1}, w_{k-2} \rangle}{\langle w_{k-2}, w_{k-2} \rangle} \langle w_{k-2}, w_j \rangle. \end{aligned} \quad (2.11)$$

Furthermore,

$$\langle w_k, w_j \rangle = \langle Aw_{k-1}, w_j \rangle + 0 + 0 = \langle w_{k-1}, Aw_j \rangle, \quad (2.12)$$

since $j < k - 2 < k - 1 < k$, by induction assumption we have $\langle w_i, w_j \rangle = 0$ for all $i \neq j < k$, and by the assumption of the algorithm A is self-adjoint relative to $\langle u, v \rangle$. Therefore, by substituting for Aw_j using (2.6), the last equation can be continued as:

$$\begin{aligned} \langle w_{k-1}, Aw_j \rangle = \langle w_{k-1}, w_{j+1} \rangle + \frac{\langle Aw_j, w_j \rangle}{\langle w_j, w_j \rangle} \langle w_{k-1}, w_j \rangle \\ + \frac{\langle Aw_j, w_{j-1} \rangle}{\langle w_{j-1}, w_{j-1} \rangle} \langle w_{k-1}, w_{j-1} \rangle, \end{aligned} \quad (2.13)$$

omitting the last term if $j = 0$. In any case,

$$\langle w_k, w_j \rangle = \langle w_{k-1}, Aw_j \rangle = 0, \quad (2.14)$$

since $j < k - 2$, implies that all the inner products on the right side of (2.13) are zero.

(2) Let $j = k - 2$. Then $\langle w_k, w_j \rangle = \langle w_k, w_{k-2} \rangle$, and by (2.6) we have:

$$\begin{aligned} \langle w_k, w_{k-2} \rangle &= \langle Aw_{k-1}, w_{k-2} \rangle - \frac{\langle Aw_{k-1}, w_{k-1} \rangle}{\langle w_{k-1}, w_{k-1} \rangle} \langle w_{k-1}, w_{k-2} \rangle \\ &\quad - \frac{\langle Aw_{k-1}, w_{k-2} \rangle}{\langle w_{k-2}, w_{k-2} \rangle} \langle w_{k-2}, w_{k-2} \rangle. \end{aligned} \quad (2.15)$$

Since by induction, $\langle w_{k-1}, w_{k-2} \rangle = 0$ and $\langle w_{k-2}, w_{k-2} \rangle \neq 0$, the last equation implies that

$$\langle w_k, w_j \rangle = \langle Aw_{k-1}, w_{k-2} \rangle - \langle Aw_{k-1}, w_{k-2} \rangle = 0. \quad (2.16)$$

(3) The case $j = k - 1$ is essentially the same as $j = k - 2$.

Therefore, $\langle w_k, w_j \rangle = 0$ is true for every $k \neq j$ and all $k \leq m$ (since k was arbitrary). Furthermore, we see that the set $\{w_0, w_1, \dots, w_{m-1}\}$ in the Lanczos Algorithm is non-isotropic and orthogonal, since $w_0 = b \neq 0$, and any other vector produced by (2.6) is non-isotropic, otherwise the algorithm stops. Finally, by Theorem 1.2.4, we know any non-isotropic orthogonal set is linearly independent. $\square$

Theorem 2.2.6. *Assume the Lanczos algorithm stops at $\langle w_m, w_m \rangle = 0$. Then:*

- (1) *For any integer $1 \leq k \leq m$, the orthogonal set $W_k = \{w_0 = b, w_1, \dots, w_{k-1}\}$ created in this algorithm is a basis for $K(A, b, k)$.*
- (2) *The set $W_m = \{w_0 = b, w_1, \dots, w_{m-1}\}$ is a basis for $K(A, b)$ if and only if $w_m = 0$, i.e., if and only if the Lanczos algorithm succeeds.*

Proof of part 1. In Theorem 2.2.5 we showed that the set W_k for $1 \leq k \leq m$ is linearly independent, so it remains to show that $K(A, b, k) = \text{Span } W_k$. Proceeding by induction on k , it is obvious that $\text{Span}\{w_0 = b\} \subseteq K(A, b, 1)$. Assume

$$\text{Span}\{w_0 = b, w_1, \dots, w_{k-1}\} \subseteq K(A, b, k) \quad (2.17)$$

for some k such that $1 \leq k < m$. From this, we can write

$$w_{k-1} = \sum_{i=0}^{k-1} \alpha_i A^i b. \quad (2.18)$$

Multiplying both sides by A we get $Aw_{k-1} = \sum_{i=0}^{k-1} \alpha_i A^{i+1} b$. This implies that $Aw_{k-1} \in K(A, b, k+1)$, and by Lanczos recursion rule (2.6) we have

$$w_k = Aw_{k-1} - \frac{\langle Aw_{k-1}, w_{k-1} \rangle}{\langle w_{k-1}, w_{k-1} \rangle} w_{k-1} - \frac{\langle Aw_{k-1}, w_{k-2} \rangle}{\langle w_{k-2}, w_{k-2} \rangle} w_{k-2}, \quad (2.19)$$

implying that $w_k \in K(A, b, k+1)$ and $\text{Span } W_{k+1} \subseteq K(A, b, k+1)$.

Conversely, again proceeding by induction on k as the induction basis it is easy to see $K(A, b, 1) \subseteq \text{Span}\{w_0 = b\}$. Assume $K(A, b, k) \subseteq \text{Span } W_k$ for some k such that $1 \leq k < m$. From this we have

$$A^{k-1}b = \beta_0 w_0 + \beta_1 w_1 + \dots + \beta_{k-1} w_{k-1}, \quad (2.20)$$

for some $\beta_i \in F$. Multiplying both sides by A gives us

$$A^k b = \beta_0 Aw_0 + \beta_1 Aw_1 + \dots + \beta_{k-1} Aw_{k-1}, \quad (2.21)$$

By Lanczos recursion rule (2.6) every Aw_j is a linear combination of $\{w_{j-1}, w_j, w_{j+1}\}$, so by substituting for each Aw_j , for some $\gamma_i \in F$ we will have

$$A^k b = \gamma_0 w_0 + \gamma_1 w_1 + \dots + \gamma_{k-1} w_{k-1} + \gamma_k w_k, \quad (2.22)$$

implying that $A^k b \in \text{Span } W_{k+1}$ and $K(A, b, k+1) \subseteq \text{Span } W_{k+1}$. Therefore,

$$K(A, b, k+1) = \text{Span } W_{k+1} \quad (2.23)$$

and the set W_{k+1} is a basis for $K(A, b, k+1)$. $\square$

Proof of part 2. Assume that $w_m = 0$. By Lanczos recursion rule (2.6) we have

$$w_m = Aw_{m-1} - \frac{\langle Aw_{m-1}, w_{m-1} \rangle}{\langle w_{m-1}, w_{m-1} \rangle} w_{m-1} - \frac{\langle Aw_{m-1}, w_{m-2} \rangle}{\langle w_{m-2}, w_{m-2} \rangle} w_{m-2} = 0, \quad (2.24)$$

implying that

$$Aw_{m-1} = \frac{\langle Aw_{m-1}, w_{m-1} \rangle}{\langle w_{m-1}, w_{m-1} \rangle} w_{m-1} + \frac{\langle Aw_{m-1}, w_{m-2} \rangle}{\langle w_{m-2}, w_{m-2} \rangle} w_{m-2} \in \text{Span } W_m. \quad (2.25)$$

This implies that $Aw_{m-1} \in K(A, b, m)$, since by part (1), $K(A, b, m) = \text{Span } W_m$.

Again by part (1) $A^{m-1}b$ is in the form of

$$A^{m-1}b = \beta_0 w_0 + \beta_1 w_1 + \dots + \beta_{m-1} w_{m-1}, \quad (2.26)$$

so

$$A^m b = \beta_0 Aw_0 + \beta_1 Aw_1 + \dots + \beta_{m-2} Aw_{m-2} + \beta_{m-1} Aw_{m-1}. \quad (2.27)$$

By substituting for Aw_{m-1} from the assumption $w_m = 0$,

$$\begin{aligned} A^m b &= \beta_0 Aw_0 + \beta_1 Aw_1 + \dots + \beta_{m-2} Aw_{m-2} \\ &\quad + \beta_{m-1} \left(\frac{\langle Aw_{m-1}, w_{m-1} \rangle}{\langle w_{m-1}, w_{m-1} \rangle} w_{m-1} + \frac{\langle Aw_{m-1}, w_{m-2} \rangle}{\langle w_{m-2}, w_{m-2} \rangle} w_{m-2} \right). \end{aligned} \quad (2.28)$$

Using (2.6) and substituting for every Aw_t where $0 \leq t < m-1$, we then have

$$A^m b = \alpha_0 w_0 + \alpha_1 w_1 + \dots + \alpha_{m-2} w_{m-2} + \alpha_{m-1} w_{m-1}, \quad (2.29)$$

implying that $A^m b \in K(A, b, m)$. Therefore, by the Dependency Theorem (A.0.8) the set $\{b, Ab, \dots, A^{m-1}b, A^m b\}$ is linearly dependent, and by Theorem 1.3.4, we have $K(A, b, m) = K(A, b)$, since m is the largest integer such that $\{b, Ab, \dots, A^{m-1}b\}$ is linearly independent.

Conversely, assume the non-isotropic set $W_m = \{w_0 = b, w_1, \dots, w_{m-1}\}$ created in the Lanczos algorithm is a basis for $K(A, b)$. By Theorem 1.2.5, the bilinear form $\langle *, * \rangle$ in this algorithm is non-degenerate over $K(A, b)$. By the last statement of

Theorem 2.2.5, $\langle w_m, w_i \rangle = 0$ for all $0 \leq i < m$. This implies that w_m is orthogonal to every vector in $K(A, b)$, which means that $w_m = 0$, since the bilinear form defined on $K(A, b)$ is non-degenerate. $\square$

Corollary 2.2.7. *If $F = R$, and the bilinear form $\langle u, v \rangle$ is an inner product over R , then Lanczos algorithm always succeeds.*

Proof. By the definition of inner product, if $\langle w_m, w_m \rangle = 0$ then $w_m = 0$. $\square$

Remark 2.2.8. In practice, when $F = R$, Lanczos algorithm's output is greatly affected by round-off error; see Golub/Van Loan [GL96] pp. 473–475 and pp. 481–482.

2.3 Using the Lanczos Algorithm to Solve $Ax = b$

Consider the Generalized Lanczos Algorithm with $\langle u, v \rangle = (u, Av)$, where A is an $n \times n$ symmetric matrix, and $(x, y) = x^T y$ is the well-known dot product. By Theorem 1.1.15, A is self-adjoint relative to $\langle u, v \rangle = (u, Av)$. Following La Macchia/Odlyzko [LO], given a successful run of the Lanczos algorithm on a given nonzero $n \times 1$ column vector b , using the matrix A and $\langle u, v \rangle = (u, Av)$, we will show the vector

$$x = \sum_{i=0}^{m-1} \frac{(w_i, b)}{\langle w_i, w_i \rangle} w_i \quad (2.30)$$

satisfies $Ax = b$. Compare Coppersmith/Odlyzko/Schroeppel [COS86], La Macchia/Odlyzko [LO] and Teitelbaum [Tei98].

Lemma 2.3.1. *With $\langle u, v \rangle = (u, Av)$, if the Lanczos algorithm succeeds, then both $\langle u, v \rangle$ and (u, v) are non-degenerate on $K(A, b)$.*

Proof. By Theorem 1.2.5, $\langle u, v \rangle$ is non-degenerate on $K(A, b)$, since the Lanczos algorithm with $\langle u, v \rangle$ is successful. So for every nonzero vector $x \in K(A, b)$, there

exists a nonzero vector $y \in K(A, b)$ such that $\langle x, y \rangle \neq 0$, implying the existence of a nonzero vector $z = Ay$ in $K(A, b)$ such that $\langle x, z \rangle \neq 0$, since $K(A, b)$ is A -invariant and $\langle x, y \rangle = \langle x, Ay \rangle$. Therefore, (u, v) is also non-degenerate on $K(A, b)$. $\square$

Theorem 2.3.2. *For an $n \times n$ symmetric matrix A , a nonzero vector b in F^n , and the bilinear form $\langle u, v \rangle = (u, Av)$, if the Lanczos algorithm succeeds, then the vector*

$$x = \frac{(w_0, b)}{\langle w_0, w_0 \rangle} w_0 + \frac{(w_1, b)}{\langle w_1, w_1 \rangle} w_1 + \dots + \frac{(w_{m-1}, b)}{\langle w_{m-1}, w_{m-1} \rangle} w_{m-1}, \quad (2.31)$$

where $W = \{w_0 = b, w_1, \dots, w_{m-1}\}$ is obtained from the Lanczos algorithm, satisfies $Ax = b$.

Proof. We have already proved (in part 2 of Theorem 2.2.6) that in a successful run of the Generalized Lanczos Algorithm, we obtain an orthogonal basis for $K(A, b)$ as $W = \{w_0, w_1, \dots, w_{m-1}\}$. Since x is a linear combination of the w_i 's, and $K(A, b)$ is an A -invariant subspace, both Ax and $Ax - b \in K(A, b)$. Since the zero vector is the only vector in the non-degenerate vector space $K(A, b)$ which is orthogonal to the whole space, it will suffice to show that the vector $Ax - b$ has the same property as zero vector in $K(A, b)$, relative to the dot product (u, v) . By (2.31),

$$\begin{aligned} Ax = \frac{(w_0, b)}{(w_0, Aw_0)} Aw_0 + \frac{(w_1, b)}{(w_1, Aw_1)} Aw_1 + \dots + \frac{(w_i, b)}{(w_i, Aw_i)} Aw_i + \dots \\ + \frac{(w_{m-1}, b)}{(w_{m-1}, Aw_{m-1})} Aw_{m-1}, \end{aligned} \quad (2.32)$$

so by bilinearity we have

$$\begin{aligned} (w_i, Ax) = \frac{(w_0, b)}{(w_0, Aw_0)} (w_i, Aw_0) + \frac{(w_1, b)}{(w_1, Aw_1)} (w_i, Aw_1) + \dots \\ + \frac{(w_i, b)}{(w_i, Aw_i)} (w_i, Aw_i) + \dots + \frac{(w_{m-1}, b)}{(w_{m-1}, Aw_{m-1})} (w_i, Aw_{m-1}). \end{aligned} \quad (2.33)$$

Since W is a non-isotropic orthogonal set, $\langle w_i, w_j \rangle = (w_i, Aw_j) = 0$ for all $i \neq j$, and $\langle w_i, w_i \rangle = (w_i, Aw_i) \neq 0$ for all i , so the equation (2.33) simplifies to

$$(w_i, Ax) = (w_i, b). \quad (2.34)$$

By bilinearity, we then have $(w_i, Ax - b) = 0$ for $0 \leq i \leq m - 1$. Therefore, the vector $Ax - b \in K(A, b)$ is orthogonal to every vector in $K(A, b)$ relative to the standard dot product since w_i are a basis for $K(A, b)$, and by Lemma 2.3.1, we conclude that $Ax - b = 0$, or $Ax = b$. $\square$

Example 2.3.3. Let

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \quad (2.35)$$

Using the Lanczos algorithm, we solve the system of equations $Ax = b$.

Our inputs are the matrix A with all entries in F_2 , the given nonzero vector $b = w_0$, and the symmetric bilinear form $\langle u, v \rangle = (u, Av)$ over F_2^5 .

Applying the Lanczos algorithm step by step, and expecting a successful run of this algorithm will provide us the orthogonal set of vectors w_i . For this since

$$\langle w_0, w_0 \rangle = (w_0, Aw_0) = (1, 0, 0, 0, 0) \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = 1 \neq 0, \quad (2.36)$$

we can calculate

$$w_1 = Aw_0 - \frac{\langle Aw_0, w_0 \rangle}{\langle w_0, w_0 \rangle} w_0 = Aw_0 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \quad (2.37)$$

Since

$$\langle w_1, w_1 \rangle = (w_1, Aw_1) = (1, 1, 0, 0, 0) \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} = 1 \neq 0 \quad (2.38)$$

and $\langle Aw_1, w_1 \rangle = (Aw_1, Aw_1) = 0$, then

$$w_2 = Aw_1 - \frac{\langle Aw_1, w_1 \rangle}{\langle w_1, w_1 \rangle} w_1 - \frac{\langle Aw_1, w_0 \rangle}{\langle w_0, w_0 \rangle} w_0 = Aw_1 - w_0 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}. \quad (2.39)$$

Again $\langle w_2, w_2 \rangle = (w_2, Aw_2) = 1 \neq 0$ and we can proceed to calculate w_3 . By the Lanczos recursive equation (2.6) we have

$$w_3 = Aw_2 - \frac{\langle Aw_2, w_2 \rangle}{\langle w_2, w_2 \rangle} w_2 - \frac{\langle Aw_2, w_1 \rangle}{\langle w_1, w_1 \rangle} w_1, \quad (2.40)$$

where $\langle Aw_2, w_2 \rangle = \langle Aw_2, w_1 \rangle = \langle w_2, w_2 \rangle = \langle w_1, w_1 \rangle = 1$. This implies that

$w_3 = Aw_2 - w_2 - w_1$. Finally,

$$w_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = 0 \quad (2.41)$$

and the algorithm successfully stops. By Theorem 2.31 using these vectors we can calculate the vector x as

$$x = \frac{(w_0, w_0)}{\langle w_0, w_0 \rangle} w_0 + \frac{(w_1, w_0)}{\langle w_1, w_1 \rangle} w_1 + \frac{(w_2, w_0)}{\langle w_2, w_2 \rangle} w_2. \quad (2.42)$$

By previous calculations and further dot product calculations each term has a unit coefficient, implying that

$$x = w_0 + w_1 + w_2 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}. \quad (2.43)$$

A direct calculation shows this is a solution for the given system of equations $Ax = b$.

2.4 Tridiagonalization by the Lanczos Algorithm

The Lanczos Algorithm can also be used as an interesting method for solving eigenvalue problems. Specifically, the Lanczos Algorithm is helpful in changing a positive definite matrix to a (possibly smaller) tridiagonal matrix with some of the same eigenvalues. It is then easier to solve the eigenvalue problem using this tridiagonal matrix. In this section, we follow Golub/Van Loan [GL96] (p. 156, and Ch. 9) and Lanczos [Lan50].

Theorem 2.4.1. *Let $V = F^n$, and let $L : V \rightarrow V$ be a linear transformation with an $n \times n$ symmetric matrix representation A , where $L(x) = Ax$ for any $n \times 1$ column vector $x \in V$. For a given nonzero vector $b \in V$, let $W = K(A, b)$, and $L_W : W \rightarrow W$ be the transformation L restricted to W . If $\alpha = \{w_0 = b, w_1, \dots, w_{m-1}\}$ is the basis for W created by a successful run of the Lanczos algorithm, then the matrix representation for L_W relative to the basis α is in tridiagonal form.*

Proof. Since W is L -invariant, we can define $T : W \rightarrow W$ by $T(x) = L(x)$, hence $[T]_\alpha$ is the matrix representation for T . By the definition of matrix representation we have $[T]_\alpha = [(T(w_0))_\alpha, (T(w_1))_\alpha, \dots, (T(w_{m-1}))_\alpha]$, where $T(w_i) = Aw_i$. Now we know $Aw_0 = \frac{\langle Aw_0, w_0 \rangle}{\langle w_0, w_0 \rangle} w_0 + w_1$, and by Lanczos recursion rule (2.6),

$$T(w_i) = Aw_i = \frac{\langle Aw_i, w_{i-1} \rangle}{\langle w_{i-1}, w_{i-1} \rangle} w_{i-1} + \frac{\langle Aw_i, w_i \rangle}{\langle w_i, w_i \rangle} w_i + w_{i+1}, \quad (2.44)$$

where $w_m = 0$. Therefore $[T]_\alpha$ has the tridiagonal form

$$[T]_\alpha = \begin{pmatrix} t_{00} & t_{01} & 0 & 0 & \dots & 0 \\ 1 & t_{11} & t_{12} & 0 & \dots & 0 \\ 0 & 1 & t_{22} & t_{23} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & t_{(m-2)(m-2)} & t_{(m-2)(m-1)} \\ 0 & 0 & \dots & \dots & 1 & t_{(m-1)(m-1)} \end{pmatrix}, \quad (2.45)$$

where

$$t_{ij} = \begin{cases} \frac{\langle Aw_j, w_{j-1} \rangle}{\langle w_{j-1}, w_{j-1} \rangle} & \text{if } i = j - 1; \\ \frac{\langle Aw_j, w_j \rangle}{\langle w_j, w_j \rangle} & \text{if } i = j; \\ 1 & \text{if } i = j + 1; \\ 0 & \text{otherwise.} \end{cases} \quad (2.46)$$

□

Remark 2.4.2. In Theorem 2.4.1, it is true that every eigenvalue of $[T]_\alpha$ is an eigenvalue of A . This is a useful fact since the tridiagonal matrix $[T]_\alpha$ is more practical for computations.

2.5 Tridiagonalization by Normalized Lanczos Vectors

We again follow Golub/Van Loan [GL96] pp. 472–473 for the material in this section.

Theorem 2.5.1. *Let $F = \mathbb{R}$ and $\langle u, v \rangle = (u, v) = u^T v$, let A be an $n \times n$ positive definite symmetric matrix, and let b be a nonzero $n \times 1$ column vector. Let the set $\alpha = \{w_0 = b, w_1, \dots, w_{m-1}\}$ be the basis created by the Lanczos algorithm for the Krylov subspace $K(A, b) = \{b, Ab, A^2b, \dots\}$. Define the set $\gamma = \{v_0, v_1, \dots, v_{m-1}\}$ such that $v_i = \frac{w_i}{|w_i|}$ and define T as in Theorem 2.4.1. Then the matrix transformation $[T]_\gamma = [(Av_0)_\gamma, (Av_1)_\gamma, \dots, (Av_{m-1})_\gamma]$ is a symmetric tridiagonal matrix.*

Proof. We know γ is an orthonormal basis for $W = K(A, b)$, since α is orthogonal.

From $w_0 = |w_0|v_0$, $v_1 = \frac{w_1}{|w_1|}$, and $w_1 = Aw_0 - \frac{\langle Aw_0, w_0 \rangle}{\langle w_0, w_0 \rangle} w_0$, we can write:

$$Av_0 = \frac{\langle Aw_0, w_0 \rangle}{\langle w_0, w_0 \rangle} v_0 + \frac{|w_1|}{|w_0|} v_1, \quad (2.47)$$

or

$$Av_0 = \beta_{00}v_0 + \beta_{10}v_1, \quad (2.48)$$

for some β_{00}, β_{10} . By the Lanczos recursive rule (2.6) we have

$$w_{i+1} = Aw_i - \frac{\langle Aw_i, w_{i-1} \rangle}{\langle w_{i-1}, w_{i-1} \rangle} w_{i-1} - \frac{\langle Aw_i, w_i \rangle}{\langle w_i, w_i \rangle} w_i. \quad (2.49)$$

Substituting for every $j \leq i+1$, $w_j = |w_j|v_j$, we have

$$Av_i = \beta_{(i-1)i}v_{i-1} + \beta_{ii}v_i + \beta_{(i+1)i}v_{i+1} \quad (2.50)$$

for some $\beta_{(i-1)i}, \beta_{ii}, \beta_{(i+1)i}$.

Since A is the matrix representation for the linear transformation $L : V \rightarrow V$, the matrix representation for $L_W : W \rightarrow W$ relative to the basis γ for $W = K(A, b)$ is $[T]_\gamma$, where

$$[T]_\gamma = \begin{pmatrix} \beta_{00} & \beta_{01} & 0 & \dots & 0 \\ \beta_{10} & \beta_{11} & \beta_{12} & \dots & 0 \\ 0 & \beta_{21} & \beta_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \beta_{(m-2)(m-1)} \\ 0 & 0 & \dots & \beta_{(m-1)(m-2)} & \beta_{(m-1)(m-1)} \end{pmatrix}. \quad (2.51)$$

By definition of $[T]_\gamma$, we have $AQ = QT$, where $Q = [v_0, v_1, \dots, v_{m-1}]$. Since $Q^T Q = I_m$ we can write $Q^T A Q = T$. Therefore, since A is symmetric, T is symmetric tridiagonal. $\square$

2.6 Normalized Lanczos Algorithm

For $F = \mathbb{R}$, and $\langle u, v \rangle = (u, v) = u^T v$, in theory, the Lanczos algorithm always succeeds and produces an orthogonal basis. Using this basis, we can reduce the given matrix A to the form of a tridiagonal matrix. By normalizing this basis, A can be reduced to a symmetric tridiagonal matrix. Putting this all together, we can write a new algorithm to get all these results at the same time. This version of the Lanczos algorithm, presented by Golub/Van Loan [GL96] p. 473 and Saad [Saa03] p. 189, is actually a more standard modern version of the Lanczos algorithm.

Algorithm 2.6.1. Normalized Lanczos algorithm

Input : An n -by- n symmetric matrix A and a nonzero n -by-1 column vector b .

Procedure : Let $q_{-1} = 0$, $q_0 = b/|b|$, and for $k \geq 1$, calculate

$$\langle q_{k-1}, Aq_{k-1} \rangle = q_{k-1}^T Aq_{k-1}, \quad (2.52)$$

$$r_k = Aq_{k-1} - \langle q_{k-1}, Aq_{k-1} \rangle q_{k-1} - \langle q_{k-2}, Aq_{k-1} \rangle q_{k-2}, \quad (2.53)$$

and if $|r_k| \neq 0$,

$$q_k = r_k/|r_k|. \quad (2.54)$$

Stop : Algorithm stops when $|r_k| = 0$.

Output : The orthonormal set $\{q_0, q_1, \dots, q_{m-1}\}$, and the entries of a tridiagonal matrix T (which can be obtained from 2 last steps defined in the algorithm's procedure).

Note that to save memory, if we want, the computer needs to store only a fixed number of q 's and the entries of T . See Golub/Van Loan [GL96], p. 473.

Theorem 2.6.2. *For an n -by- n symmetric matrix A , and a nonzero n -by-1 column vector b , the vectors produced in the Normalized Lanczos Algorithm are the vectors $\gamma = \{v_0, v_1, \dots, v_{m-1}\}$ obtained by normalizing the output of the Generalized Lanczos Algorithm.*

Proof. Define the symmetric bilinear form $\langle u, v \rangle = u^T v$. Let $\{w_0 = b, w_1, \dots, w_{m-1}\}$ be the output of the Generalized Lanczos Algorithm. We name the normal output vectors from the Normalized and Generalized Lanczos Algorithms respectively as $q_k = r_k/|r_k|$ and $v_k = w_k/|w_k|$, where

$$r_k = Aq_{k-1} - \langle Aq_{k-1}, q_{k-1} \rangle q_{k-1} - \langle Aq_{k-1}, q_{k-2} \rangle q_{k-2}, \quad (2.55)$$

as above. Recall that $w_0 = b$, $w_1 = Aw_0 - \frac{\langle Aw_0, w_0 \rangle}{\langle w_0, w_0 \rangle} w_0$, and by recursive formula equation (2.6),

$$w_k = Aw_{k-1} - \frac{\langle Aw_{k-1}, w_{k-1} \rangle}{\langle w_{k-1}, w_{k-1} \rangle} w_{k-1} - \frac{\langle Aw_{k-1}, w_{k-2} \rangle}{\langle w_{k-2}, w_{k-2} \rangle} w_{k-2}. \quad (2.56)$$

It is obvious that $q_0 = v_0 = b/|b|$. Proceeding by induction on k , we will prove $v_k = w_k/|w_k| = r_k/|r_k| = q_k$ for all $k > 0$. From $v_0 = w_0/|w_0|$ we have $w_0 = |w_0|v_0$, so by substituting w_0 in the equation $w_1 = Aw_0 - \frac{\langle Aw_0, w_0 \rangle}{\langle w_0, w_0 \rangle} w_0$, we have

$$\begin{aligned} w_1 &= |w_0|Av_0 - \frac{|w_0|^2 \langle Av_0, v_0 \rangle}{|w_0|^2 \langle v_0, v_0 \rangle} |w_0|v_0 \\ &= |w_0|(Av_0 - \frac{\langle Av_0, v_0 \rangle}{\langle v_0, v_0 \rangle} v_0) \\ &= |w_0|(Av_0 - \langle Av_0, v_0 \rangle v_0) = |w_0|r_1, \end{aligned} \tag{2.57}$$

since $|v_0| = |q_0| = 1$. Therefore

$$v_1 = w_1/|w_1| = |w_0|r_1/||w_0|r_1| = r_1/|r_1| = q_1. \tag{2.58}$$

Now by induction, assume $v_i = q_i$ for all $i < k$. Then by substituting for vectors $w_{k-1} = |w_{k-1}|v_{k-1}$ and $w_{k-2} = |w_{k-2}|v_{k-2}$ in (2.6), by bilinearity we have

$$\begin{aligned} w_k &= |w_{k-1}|Av_{k-1} - \frac{|w_{k-1}|^2 \langle Av_{k-1}, v_{k-1} \rangle}{|w_{k-1}|^2 \langle v_{k-1}, v_{k-1} \rangle} |w_{k-1}|v_{k-1} \\ &\quad - \frac{|w_{k-1}||w_{k-2}| \langle Av_{k-1}, v_{k-2} \rangle}{|w_{k-2}|^2 \langle v_{k-2}, v_{k-2} \rangle} |w_{k-2}|v_{k-2}. \end{aligned} \tag{2.59}$$

After simplifying the fractions we have

$$w_k = |w_{k-1}|(Av_{k-1} - \langle Av_{k-1}, v_{k-1} \rangle v_{k-1} - \langle Av_{k-1}, v_{k-2} \rangle v_{k-2}), \tag{2.60}$$

since $|v_{k-2}|^2 = |v_{k-1}|^2 = 1$.

Therefore, since $r_k = Av_{k-1} - \langle Av_{k-1}, v_{k-1} \rangle v_{k-1} - \langle Av_{k-1}, v_{k-2} \rangle v_{k-2}$, we see that $w_k = |w_{k-1}|r_k$, leading to the conclusion that

$$v_k = w_k/|w_k| = \frac{|w_{k-1}|r_k}{||w_{k-1}|r_k|} = r_k/|r_k| = q_k. \tag{2.61}$$

□

CHAPTER 3

NUMBER THEORY

3.1 Introduction

A prime number is a number with only one positive divisor other than 1. If a number is not prime, then it should have more than two positive divisors with some primes among them. For a nonzero integer m , the form $m = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$ where each p_i is a distinct prime number is known as the *Prime Factorization* of m .

One recent application of prime factorization is in the area of *cryptology*, or encoding and decoding secretive messages. There are many different ways of coding and decoding secretive messages or identifications. Since computers have become popular and more convenient than any other communication, trade, and information-exchange device, security is a major concern for both the information provider and the user. When engineers started to create security codes, they tried their best to invent codes that would be as difficult as possible to break (unbreakable). This is a proof for the old saying that “Necessity is the mother of invention.”

One important invention is known as *RSA Public Key Cryptography*, which is based on prime factorization. The security of RSA requires that large numbers are hard to factor. This encoding and one-way decoding system which is called *Public Key Cryptography* is a good reason to know all about the Lanczos algorithm and its applications.

First let us review some standard theorems and definitions in number theory.

Definition 3.1.1. [Sil01] p. 65, *Euler's phi function*: The number of integers between 0 and m that are relatively prime to m is called Euler's phi function $\phi(m)$. This can also be written as:

$$\phi(m) = \#\{a : 1 \leq a \leq m \text{ and } \gcd(a, m) = 1\}. \quad (3.1)$$

Notice that if p is prime, then $\phi(p) = p - 1$.

Theorem 3.1.2. [Sil01] *Euler's Theorem*: For nonzero integers a and m , if the $\gcd(a, m) = 1$, then $a^{\phi(m)} \equiv 1 \pmod{m}$.

Proof. See Silverman [Sil01], Chapter 10, p. 66. □

Theorem 3.1.3. [Sil01] *Euler's phi function is a multiplicative function, which means if $\gcd(m, n) = 1$, then $\phi(mn) = \phi(m)\phi(n)$.*

Proof. See Silverman [Sil01], Chapter 11, p. 70. In particular if p and q are primes then $\phi(pq) = (p - 1)(q - 1)$. □

3.2 RSA Cryptography

This section is based on material from Silverman [Sil01] Ch. 18, and Bressoud [Bre89] Ch. 4. In late 70's Ron Rivest, Adi Shamir and Len Adleman brought convenience to the security industry through their brilliant schemes that relied on secrecy just for decoding; see Crandall/Pomerance [CP05] p. 2. The idea is called Public Key Cryptography.

In Public Key Cryptography only the receiver has access to the decoding secret key, while everybody on the earth is able to encode their secret message to the receiver. To illustrate RSA, we need two secretly known prime numbers like p and q , and two

publicly disclosed numbers m and k , where $m = pq$ and k is any arbitrary number such that $\gcd(k, \phi(m))=1$.

Example 3.2.1. Susan writes $m=8051$ and $k = 15377$ on the board. She asks each student to choose any positive number a less than m as his/her secret message. She asks them to calculate $b \equiv a^k \pmod{m}$ and disclose their number b on the board. Within seconds she writes the student's chosen secret message a , since she is the only person who knows the secret code d for decoding student's number. More precisely Tom discloses his encoded number $b = 27$, and Susan decodes the message as $a = 1870$. As long as she is the only person in the world who knows the secret number d , no one else can decode students' secret messages.

Algorithm 3.2.2. RSA encoding algorithm (Tom)

Input : The public numbers m and k (for example $m = 8051$ and $k = 15377$), where k is relatively prime to $\phi(m)$, and any message as an integer $a < m$.

Procedure : Compute $a^k \equiv b \pmod{m}$.

Output : Send the encoded message b .

Algorithm 3.2.3. RSA decoding algorithm (Susan)

Input : The secret number d (for example $d = 5105$), where $dk \equiv 1 \pmod{\phi(m)}$, the public number m , and the encoded message b .

Procedure : Compute $b^d \pmod{m}$.

Output : The decoded message a .

Theorem 3.2.4. *In the RSA decoding procedure, calculating b^d modulo m provides the original encoded message a .*

Proof. Since $b \equiv a^k \pmod{m}$, we can write $b^d \equiv (a^k)^d \pmod{m}$. This implies that for some $t \in \mathbb{Z}$,

$$b^d \equiv (a)(a^t)^{\phi(m)} \pmod{m}, \quad (3.2)$$

because $dk \equiv 1 \pmod{\phi(m)}$. Since $\gcd(a, m) = 1$, by Euler's theorem we have $a^{\phi(m)} \equiv 1 \pmod{m}$. Therefore, $b^d \equiv a \pmod{m}$. $\square$

3.3 Factoring

What makes the RSA cryptosystem secure is the complexity of factoring the public key m .

Definition 3.3.1. [CP05] p. 8, When we say $f(N)=O(g(N))$, we mean that there exists a positive number C such that for all N in a specified set,

$$|f(N)| \leq C|g(N)|. \quad (3.3)$$

See Crandall/Pomerance[CP05] p. 8.

Definition 3.3.2. [CP05] The Time Complexity of a decision problem, f , is the running time of the 'best' algorithm for f . Thus we say that a decision problem, f , has *Time Complexity* $T(n)$ if there is an algorithm (the best algorithm) for f such that the number of steps taken by any algorithm on inputs of length n (size n) is always less than or equal to $T(n)$. For further reading see Crandall/Pomerance [CP05] chapter 1, and Dunn [Dun03].

Example 3.3.3. [CP05] p. 85. Let $a > b$ be integers from the interval $[1, N]$. Then the time complexity of the Euclidean algorithm for $\gcd(a, b)$ is $O(\ln^2 N)$.

Theorem 3.3.4. *Finding the secret number d in the RSA decoding procedure is possible if we know the prime factorization of the public number m .*

Proof. From the prime factorization $m = pq$, by Theorem 3.1.3, $\phi(m) = (p-1)(q-1)$. Since we know the public number k is relatively prime to $\phi(m)$, then by solving equation $dk \equiv 1 \pmod{\phi(m)}$ for d (like solving $15377d \equiv 1 \pmod{\phi(8051)}$ for Tom), we will obtain the secret code d . (The algorithm for solving $dk \equiv 1 \pmod{\phi(m)}$ is a fast algorithm like the Euclidean algorithm; see Silverman [Sil01], p. 54.) $\square$

According to the RSA decoding procedure, what do we need to be able to decode the captured secret message is to find out what the number d is? By Theorem 3.3.4 it is enough to know the prime divisors of our publicly disclosed number m . Even for an eighth grade student answering this question looks easy as long as he has learned about Erathosthenes' sieving algorithm. However the complexity of the primality test by Erathosthenes' sieve for an entry's size n is about $O(\sqrt{n})$, meaning that this method is not fast enough to handle super giant composite numbers, when their prime divisors are giants themselves.

So here come some newly invented factorization methods, such as: Pollard's ρ and $\rho - 1$ methods, The Quadratic Sieve factorization, Lenstra's Elliptic Curve method, and The Number Field Sieve factorization. These methods were created mostly because of the importance of factoring in cryptographic security. Based on experimental observation, the time complexity for some of these algorithms are stated in computational and algorithm comparison books; for example, see Crandall/Pomerance [CP05].

3.4 Quadratic Residues and Fermat's Square Pair Factorization

As a prologue to the Quadratic Sieve algorithm let us review some standard material about integers from Ireland [IR90], Silverman [Sil01], and Crandall/Pomerance [CP05].

Definition 3.4.1. [IR90] p. 50. An integer $a \not\equiv 0 \pmod{p}$, is a *quadratic residue* (QR) modulo the prime number p if $a \equiv r^2 \pmod{p}$ for some integer r ; otherwise a is called a *quadratic non residue* (NR) mod p .

Definition 3.4.2. [IR90] p. 51. For the prime p and an integer a , where $a \not\equiv 0 \pmod{p}$, we define the Legendre Symbol (a/p) by:

$$(a/p) = \begin{cases} 1 & \text{if } a \text{ is a QR,} \\ -1 & \text{if } a \text{ is a NR.} \end{cases} \quad (3.4)$$

Theorem 3.4.3. [Quadratic Residue Multiplication Rule][Sil01] For a prime p and integers $a \not\equiv 0 \pmod{p}$ and $b \not\equiv 0 \pmod{p}$, we can show $(a/p)(b/p) = (ab/p)$.

Proof. See Silverman [Sil01], p. 148. □

Definition 3.4.4. [IR90] p. 56. For any integer a and any odd positive integer b with prime factorization $b = p_1 p_2 \dots p_m$, where the p_i s are not necessarily distinct, the Jacobi symbol $\left(\frac{a}{b}\right)$ is defined to be:

$$\left(\frac{a}{b}\right) = (a/p_1)(a/p_2) \dots (a/p_m). \quad (3.5)$$

From the definition it is obvious that for a prime number p :

$$\left(\frac{a}{p}\right) = (a/p) = \begin{cases} 1 & \text{if } a \text{ is a QR,} \\ -1 & \text{if } a \text{ is a NR,} \end{cases} \quad (3.6)$$

and by the Legendre symbol property Theorem 3.4.3 we can write $\left(\frac{a}{c}\right) \left(\frac{b}{c}\right) = \left(\frac{ab}{c}\right)$.

For further comprehensive reading see Ireland/Rosen [IR90] or Silverman [Sil01].

Theorem 3.4.5. [IR90] *General Quadratic Reciprocity Law for Jacobi symbols. For any positive odd integers m and n :*

$$\left(\frac{-1}{n}\right) = (-1)^{(n-1)/2} = \begin{cases} 1 & \text{if } n \equiv 1 \pmod{4}, \\ -1 & \text{if } n \equiv 3 \pmod{4}; \end{cases} \quad (3.7)$$

$$\left(\frac{2}{n}\right) = (-1)^{(n^2-1)/2} = \begin{cases} 1 & \text{if } n \equiv 1 \text{ or } 7 \pmod{8}, \\ -1 & \text{if } n \equiv 3 \text{ or } 5 \pmod{8}; \end{cases} \quad (3.8)$$

and

$$\left(\frac{n}{m}\right) = \begin{cases} \left(\frac{m}{n}\right) & \text{if } n \text{ or } m \equiv 1 \pmod{4}, \\ -\left(\frac{m}{n}\right) & \text{if } n \text{ and } m \equiv 3 \pmod{4}; \end{cases} \quad (3.9)$$

or in other words,

$$\left(\frac{n}{m}\right) \left(\frac{m}{n}\right) = (-1)^{(\frac{m-1}{2})(\frac{n-1}{2})}. \quad (3.10)$$

By the General Quadratic Reciprocity Law for Jacobi symbols, we do not need to know the prime factorization for each involved integers m and n to find out about its quadratic residue behavior. This makes the quadratic residue process faster and easier. The time complexity for calculating $\left(\frac{a}{m}\right)$ is $O(\ln^2 m)$ when $|a| < m$, for a given positive odd integer m and integer a ; see Crandall/Pomerance [CP05] p. 98.

The idea of the following theorem was invented by Gauss and Seelhoff, but it was Kraitchik who brought it up and made it practical centuries later; see Pomerance [Pom96]. Afterwards it was used to invent the Quadratic Sieve algorithm.

Theorem 3.4.6. *If $a^2 \equiv b^2 \pmod{m}$, where m has at least two prime divisors, and*

$$a \not\equiv \pm b \pmod{m}, \quad (3.11)$$

then we can find the nontrivial divisors of m .

Such pairs a, b as in this theorem are called the *interesting solutions* for finding non-trivial factors of the integer m ; see Pomerance [Pom96].

Proof. By assumption $m \mid a^2 - b^2$, and neither $a + b$ nor $a - b$ is divisible by m . If p is a prime divisor of m then $p \mid m$, implying that $p \mid a - b$ or $p \mid a + b$. This implies that one of

$$d = \gcd(a - b, m) \tag{3.12}$$

or

$$d' = \gcd(a + b, m). \tag{3.13}$$

is greater than 1. However, if $d \neq 1$, then $d \neq m$, since by the assumption, we know $a \not\equiv b \pmod{m}$. Therefore, there exists $\gcd(m, a - b) = d$ that could contain or be one of the prime divisors of m , which we are interested in. A similar argument applies if $p \mid a + b$. $\square$

3.5 Quadratic Sieve

The young algorithm of *Quadratic Sieve Factorization* (invented by Pomerance in 1981) is a powerful relatively new technique for factorization which is valuable today because of high security needs. Since the basic idea behind this algorithm is Fermat's square pair factorization, we continue this chapter by using this factorization to explain how the *Quadratic Sieve Factorization* algorithm works. In the next chapter we will talk about how Lanczos algorithm helps us to accelerate the procedure in the *Quadratic Sieve Factorization* method. The following section is based on Pomerance [Pom96] and Crandall/Pomerance [CP05] Ch. 6.

Definition 3.5.1. A *Factor Base* is a nonempty set of primes, as well as possibly -1 .

Definition 3.5.2. An integer is called *smooth* over a Factor Base if its unique factorization form contains only primes are elements of that Factor Base.

Algorithm 3.5.3. Quadratic Sieve Algorithm- (Crandall/Pomerance[CP05], p. 266.)

Input : Any positive integer m , and an arbitrary number B which is a function of m . (By experiment, it seems $B \approx \exp \frac{1}{2} \sqrt{\log m \log \log m}$ is the best choice for this algorithm, see Crandall/Pomerance [CP05], p. 265.)

Procedure :

- (1) Calculate $\lfloor \sqrt{m} \rfloor$.
- (2) Define the set $U_B = \{\text{all prime numbers less than or equal to } B, \text{ as well as } -1\}$.
- (3) Create a Factor Base $U_K = \{p_i \in U_B \mid \left(\frac{m}{p_i}\right) = 1\}$, where K is the number of primes in U_K .
- (4) Calculate $r_t = \lfloor \sqrt{m} \rfloor + t$ and $f(r_t) = r_t^2 - m$. For $t = 0, -1, 1, -2, 2, \dots$, and choose $N \geq K + 1$ number of $f(r_t)$ where every $f(r_t)$ is smooth over U_K . Each $f(r_t)$ being smooth over U_K is possible only because U_K contains primes p_i such that m is a quadratic residue modulo p_i .
- (5) Assign the vector $v_t = (x_1, x_2, \dots, x_K)^T \in F_2^K$ for each $f(r_t)$ factored as $f(r_t) = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_K^{\alpha_K}$, where $x_i \equiv \alpha_i \pmod{2}$.
- (6) Check the linear dependency between the defined exponential vectors in the previous step. In other words, solve $Ax = 0$ where A is a $K \times N$ matrix created from these exponential vectors in its columns.

Output : Present the solution vector $w \in F_2^N$ for $Ax = 0$. If the presented vector leads us to an interesting solution then we are done, otherwise we have to start over.

Remark 3.5.4. Creating the factor base U_K in step 3 of the algorithm will accelerate the sieving process by eliminating the primes less than B such that $\left(\frac{m}{p}\right) \neq 1$, since these primes are useless for producing Fermat's squares for factoring m . This is because if for a prime p , p divides $f(r)$, then we will have $\left(\frac{m}{p}\right) = 1$, since the construction of $f(r) = r^2 - m$ implies that $m \equiv r^2 \pmod{p}$ for some r . Therefore, if $\left(\frac{m}{p}\right) \neq 1$, then $f(r)$ is not divisible by p and no further computing with p will be useful.

Theorem 3.5.5. *The output vector of the Quadratic Sieve algorithm $w \in F_2^N$, will produce an equivalence of the form $a^2 \equiv b^2 \pmod{m}$.*

Proof. The vector $w \in F_2^N$ such that $Aw = 0$ implies that a linear combination of the vectors v_t is zero. Since the only scalars in F_2 are 0 and 1, then for some $i_1 \dots i_\gamma$, we have $v_{i_1} + \dots + v_{i_\gamma} = 0$, which translates to a multiplication of relative $f(r_t)$'s to a perfect square. Therefore, we can write

$$\begin{aligned} p_1^{\beta_1} p_2^{\beta_2} \dots p_K^{\beta_K} &= f(r_{i_1}) \cdot f(r_{i_2}) \dots f(r_{i_\gamma}) \\ &\equiv (r_{i_1})^2 (r_{i_2})^2 \dots (r_{i_\gamma})^2 \pmod{m} \end{aligned} \quad (3.14)$$

where all β_i are even to produce the zero vector in F_2^K and a perfect square in Z . Now let

$$a = p_1^{\beta_1/2} p_2^{\beta_2/2} \dots p_K^{\beta_K/2}, \quad (3.15)$$

and

$$b = r_{i_1} r_{i_2} \dots r_{i_\gamma}, \quad (3.16)$$

since $f(r_{i_t}) \equiv r_{i_t}^2 \pmod{m}$ for every $1 \leq t \leq \gamma$. Therefore, an equivalence of the form of $a^2 \equiv b^2 \pmod{m}$ is produced. $\square$

Now according to the definition of the *interesting solutions* for finding non-trivial factors of integer m , if we find no interesting solutions for $a^2 \equiv b^2 \pmod{m}$

then we could start the algorithm at step 4 by choosing a new set of $f(r_t)$'s, or at step 1 by choosing a new factor base.

According to [Pom96], the time complexity for factoring m by Quadratic Sieve algorithm is $\exp \sqrt{\log m \log \log m} \approx B^2$. To see how often this algorithm succeeds, see Crandall/Pomerance [CP05], Ch. 6, pp. 266-268.

Example 3.5.6. We factor 2041 using the Quadratic Sieve Algorithm. First we calculate $\sqrt{2041} = 45.1774$, and for arbitrary $B = 6$ we choose $U = \{2, 3, 5\}$. Then we follow the algorithm's next steps as:

- (1) For $t \geq 1$ in $r_t = \lfloor \sqrt{2041} \rfloor + t$ and substituting produced $r_t = 46, 47, 48, 49, \dots$, in the equation $f(r_t) = r_t^2 - 2041$ we will have the sequence of numbers such as:

$$f(r_1) = 75, \quad f(r_2) = 168, \quad f(r_3) = 263, \quad f(r_4) = 360, \quad (3.17)$$

$$f(r_5) = 459, \quad f(r_6) = 560, \quad f(r_7) = 663, \quad f(r_8) = 768, \quad (3.18)$$

$$f(r_9) = 875, \quad \dots, \quad f(r_{14}) = 1440, \quad \dots \quad (3.19)$$

Using the prime factorization of these non-square numbers we will choose at least $N = 4$ of the numbers which are smooth over U , since U contains $K = 3$ elements. According to the following prime factorizations we choose 75, 360, 768, and 1440.

$$\begin{array}{lll} \boxed{75 = 3 \times 5^2}, & 168 = 2^3 \times 3 \times 7, & \boxed{360 = 2^3 \times 3^2 \times 5}, \\ 560 = 2^4 \times 5 \times 7, & 663 = 3 \times 13 \times 17, & \boxed{768 = 2^8 \times 3}, \\ \boxed{1440 = 2^5 \times 3^2 \times 5}. & & \end{array}$$

- (2) Notice that for each $f(r_t)$ we have $f(r_t) \equiv r_t^2 \pmod{2041}$. So for the chosen

$f(r_t)$'s, this is

$$75 \equiv 46^2 \pmod{2041}, \quad (3.20)$$

$$360 \equiv 48^2 \pmod{2041}, \quad (3.21)$$

$$768 \equiv 53^2 \pmod{2041}, \quad (3.22)$$

$$1440 \equiv 59^2 \pmod{2041}. \quad (3.23)$$

- (3) According to the selected U , for each $f(r_t)$ that factors over U we define exponential vectors $v_t = (x_1, x_2, \dots, x_K)^T \in F_2^K$, where $x_i \equiv \alpha_i \pmod{2}$. These vectors in the current case with $U = \{2, 3, 5\}$ are

$$v_{75} = (0, 1, 0)^T, v_{360} = (1, 0, 1)^T, v_{768} = (0, 1, 0)^T, \text{ and } v_{1440} = (1, 0, 1)^T$$

- (4) Check for a linear dependency between the defined exponential vectors for the chosen $f(r_t)$.

In other words, solve $Ax = 0$, where A is a $K \times N$ matrix created from these exponential vectors in its columns. In the case of 2041 it is

$$Ax = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} x = 0. \quad (3.24)$$

Here $x = (1, 0, 1, 0)^T$ is a nontrivial solution in F_2^3 for this system of equations, meaning that $1.v_{75} + 0.v_{360} + 1.v_{768} + 0.v_{1440} = (0, 0, 0)^T$.

- (5) Translating this vector equation to the $f(r_t)$'s multiplication we have $75 \times 768 = 2^8 \times 3^2 \times 5^2$, a perfect square. Now we have found Kraitchik's desired quadratic equivalence (Theorem 3.4.6):

$$(46 \times 53)^2 \equiv (2^4 \times 3 \times 5)^2 \pmod{2041}. \quad (3.25)$$

(6) Is this an interesting solution? To answer this question, since we have

$$a = 46 \times 53 \equiv 397 \pmod{2041}, \quad (3.26)$$

$$b = 2^4 \times 3 \times 5 \equiv 240 \pmod{2041}, \quad (3.27)$$

we see that $397 \not\equiv \pm 240 \pmod{2041}$. Therefore, by Theorem 3.4.6, the $\gcd((397 - 240), 2041) = 157$ divides 2041. Finally, $2041 = 13 \times 157$.

CHAPTER 4

FERMAT MEETS LANCZOS

Finding a linear dependency between the exponential vectors in the last step of Quadratic Sieve Algorithm requires solving a gigantic homogeneous linear system of equations. For example for factoring a 100 digit number m , if $m \approx 10^{100}$ then the preferred $B \approx 48,400,000$ will create a coefficient matrix with columns in F_2^K where $K \approx 1,500,000$. This is where Lanczos comes to help his older colleagues, especially the one who initiated the idea of square pair factorization method, Fermat. In this chapter we follow D. Coppersmith, A.M. Odlyzko and R. Schroepel [COS86], La Macchia/Odlyzko [LO] and Montgomery [Mon95] to explain how this cooperation works.

4.1 Review of Algebraic Problems Required for Quadratic Sieve

At the sixth step of the Quadratic Sieve algorithm we need to find a linear dependency between the produced exponential vectors. This means we have to find a nonzero solution for $Ax = 0$ in F_2^N , where A is a $K \times N$ matrix with each column presenting an exponential vector with entries from F_2 and $N > K$. Solving this system in F_2 eliminates the round-off error problem that mentioned in chapter 2.1, but there are some others that prevent the use of the Lanczos algorithm. First, the system of equations is homogeneous, and second, the coefficient matrix is not square.

Finally, since the chosen field is finite (F_2) and the coefficient matrix A produced in the algorithm is sparse, there is a high possibility of self-conjugacy ($\langle w_i, w_i \rangle = 0$, $w_i \neq 0$). Following Coppersmith/Odlyzko/Schroeppel [COS86], La Macchia/Odlyzko [LO] and Montgomery [Mon95], in the next sections, we explain how these problems can be overcome.

4.2 Reducing $Ax = 0$ to $Ax = b \neq 0$

In the above situation, we have $Ax = 0$, implying that the Krylov subspace method previously discussed for $Ax = b$, where $b \neq 0$, is not applicable. But another interpretation of finding a nonzero solution for $Ax = 0$ is to show that there is a linear dependency between some of the columns of A . In other words, we need to show now we can write at least one of its columns as a linear combination of the others. (See Montgomery [Mon95], pp. 113–114.)

Following Montgomery, let the vector $a_i = (a_{1i}, a_{2i}, \dots, a_{Ki})^T$ be the i th column of the $K \times N$ matrix A in $Ax = 0$. To see if this arbitrary column vector is a linear combination of the others, we need to solve the system of equations $A_i y = a_i$, where A_i is the $K \times (N - 1)$ matrix with the same columns as matrix A except with the i th column omitted. Now we have created a non-homogeneous system of equations which might be solvable by Lanczos methods according to the next sections. Let $y = (y_1, y_2, \dots, y_{N-1})^T$ be a solution for the system of equations $A_i y = a_i$. It is easy to see that $Y = (y_1, y_2, \dots, y_{i-1}, 0, y_i, \dots, y_{N-1})^T$ is a solution for the system of equations $Ax = a_i$. Since $Z = (0, 0, \dots, 0_{i-1}, 1, 0_{i+1}, \dots, 0)^T$ is also a solution for $Ax = a_i$, we have found at least two distinct solutions for $Ax = a_i$, namely, Y and Z . Therefore, the vector $x = Y - Z$ is a nontrivial solution for $Ax = 0$, since $AY = a_i$ and $AZ = a_i$.

If the selected vector a_i doesn't lead us to a nontrivial solution for $Ax = 0$, then we can choose another column vector and try it again. Since in Quadratic Sieving method the number of vectors (N) is greater than the number of primes in U_K (namely K), finding such dependent column vector is inevitable.

4.3 Reducing $Ax = b$ for Non-Symmetric A to $Bx = c$ for Symmetric B

In the equation $Ax = b$, the matrix A obtained above is not square. By multiplying both sides by A^T we produce the system of equations $A^T Ax = A^T b$ or $Bx = c$, where B is symmetric and has the same rank as matrix A . (See Copper-smith/Odlyzko/Schroeppel [COS86], and La Macchia/Odlyzko[LO].)

Theorem 4.3.1. *Consider $Ax = b$, where A is a $K \times N$ ($K \leq N$) matrix and b an $N \times 1$ nonzero column vector.*

- (1) *Every solution for the system of equations $Ax = b$ is a solution for the system of equations $A^T Ax = A^T b$.*
- (2) *If A has full rank, then every solution for $A^T Ax = A^T b$ is a solution for the system of equations $Ax = b$.*

Proof of part 1. Let y be a solution for $Ax = b$. Then $Ay = b$. By multiplying A^T on both sides we have $A^T Ay = A^T b$. □

Proof of part 2. Let z be a vector such that $A^T Az = A^T b$. Then $A^T(Az - b) = 0$, implying that $Az - b \in \text{Null}(A^T)$. By assumption $\text{rank}(A^T) = \text{rank}(A) = K$. Applying the corollary to Theorem A.0.4 for matrix A^T we can write

$$\text{rank}(A^T) + \text{nullity}(A^T) = K, \tag{4.1}$$

which brings us to conclusion that $Az - b = 0$. Therefore, $Az = b$. □

Remark 4.3.2. In part 2 of the Theorem 4.3.1 if A is not a full rank matrix, then the conclusion may not be true. However, experiments show that even if A does not have full rank, then still there is a high possibility of correctness for this part of the theorem as well. See La Macchia/Odlyzko[LO].

4.4 Avoiding Self-Conjugacy

After we have reduced $Ax = 0$ to $Ax = b \neq 0$, and know how to solve the matrix symmetrization problem required in Lanczos algorithm, the last barrier in finite fields is how to overcome self-conjugacy ($\langle w_m, w_m \rangle = 0, w_m \neq 0$) at the stopping point in the algorithm (Theorem 2.2.6).

Following Coppersmith/Odlyzko/Schroeppel [COS86], if the field F is large enough, instead of $Ax = b$ we can solve $DAx = Db$, where D is a diagonal matrix whose diagonal elements are chosen at random from $F - \{0\}$. Now using the previously described method on new system of equations $DAx = Db$ we can apply Lanczos algorithm on the N -by- N symmetric matrix $B = A^T D^2 A$ using the N -by-1 column vector $c = A^T D^2 b$. By choosing D at random, we hope that B is as close to full rank as possible. This means that hopefully solutions for $A^T D^2 Ax = A^T D^2 b$ are also solutions of $DAx = Db$. Experimentally (Coppersmith/Odlyzko/Schroeppel [COS86]) the value of $\langle w_i, Bw_i \rangle$ will be zero about $\frac{1}{|F|}$ of the times. Then as long as $|F| > \text{rank}(A)$, we expect algorithm to succeed. If $|F| \leq \text{rank}(A)$, for example $F = GF(2)$, then we embed F in a larger field like $GF(2^r)$, for some r such that $2^r > \text{rank}(A)$, and we try the algorithm again.

Remark 4.4.1. Consider the symmetric $N \times N$ matrix $B = A^T D^2 A$ and $N \times 1$ column vector $c = A^T D^2 b$, where A is an $K \times N$ matrix, D an $N \times N$ diagonal and b an $N \times 1$ column vector. Solving the system of equations $Bx = c$ gives us a solution

such as $y = (y_1, y_2, \dots, y_N)^T$ with $y_i \in GF(2^r)$, where y is often also a solution for $Ax = b$ (for example if B has full rank). However, we need a solution with entries in $GF(2)$ for $Ax = b$.

By Theorem A.0.9, $GF(2^r) \cong GF(2)[t]/(p(t))$, where $p(t)$ is an irreducible polynomial of degree r over $GF(2)$. This implies that elements of $GF(2^r)$ can be written uniquely as polynomials of degree less than r with coefficients in $GF(2)$. So for each y_i we have

$$y_i = f_i(t) = \alpha_{i_{r-1}}t^{r-1} + \dots + \alpha_{i_1}t + \alpha_{i_0}, \quad (4.2)$$

where t is indeterminate and $\alpha_{i_j} \in GF(2)$. The solution y for $Ax = b$ can be illustrated as

$$Ay = \begin{pmatrix} a_{11} & a_{12} & \dots & \dots & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & \dots & \dots & a_{2N} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{K1} & a_{K2} & \dots & \dots & \dots & a_{KN} \end{pmatrix} \begin{pmatrix} f_1(t) \\ f_2(t) \\ \dots \\ \dots \\ f_N(t) \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ \dots \\ b_K \end{pmatrix} = b, \quad (4.3)$$

where a_{ij} and b_i are in $GF(2)$. Since the above matrix equation never uses the fact that $p(t) = 0$ in $GF(2)$, by plugging $t = 0$ in each $f_i(t)$, we see that the vector $(f_1(0), f_2(0), \dots, f_N(0))^T$ is a solution in $GF(2)$ for the system of equations $Ax = b$.

BIBLIOGRAPHY

- [Art91] M. Artin, *Algebra*, Prentice-Hall, 1991.
- [Bre89] David M. Bressoud, *Factorization and primality testing*, Springer-Verlag, 1989.
- [Bri98] Mathew E. Briggs, *An introduction to the general number field sieve*, Master's Thesis, Virginia Polytechnic Institute and State University, 1998.
- [COS86] D. Coppersmith, A.M. Odlyzko, and R. Schroeppe, *Discrete logarithms in $GF(p)$* , *Algorithmica* **1** (1986), no. 1, 1–15.
- [CP05] Richard Crandall and Carl Pomerance, *Prime numbers: A computational perspective*, Springer, 2005.
- [Dun03] Paul E. Dunn, *Computability and complexity — overview*, unpublished, www.csc.liv.ac.uk/~ped/teachadmin/algor/comput_complete.html, 2003.
- [FIS97] Stephen H. Friedberg, Arnold J. Insel, and Lawrence E. Spence, *Linear algebra*, Prentice-Hall, 1997.
- [GL96] G. Golub and C.F. Van Loan, *Matrix computation*, Johns Hopkins, 1996.
- [IM] Isle C.F. Ipsen and Carl D. Meyer, *The idea behind Krylov methods*, unpublished, citeseer.ist.psu.edu/135899.html.
- [IR90] Kenneth Ireland and Michael Rosen, *A classical introduction to modern number theory*, Springer, 1990.
- [Lan50] Cornelius Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, *Journal of Research of the National Bureau of Standards* **45** (1950), 255–282.
- [LO] B. A. LaMacchia and A.M. Odlyzko, *Solving large sparse linear systems over finite fields*, unpublished, www.crypto-world.com/FactorPapers.html.

- [Mon95] Peter L. Montgomery, *A block Lanczos algorithm for finding dependencies over $GF(2)$* , Lecture Notes in Computer Science Advances in Cryptology-EUROCRYPT '95 **921** (1995), 106–120, Springer.
- [Pis84] Sergio Pissanetsky, *Sparse matrix technology*, Academic Press, 1984.
- [Pom96] Carl Pomerance, *A tale of two sieves*, Notices of the AMS **43** (1996), no. 12, 1473–1485.
- [Saa03] Yousef Saad, *Iterative methods for sparse linear systems*, SIAM, 2003.
- [Sil01] Joseph H. Silverman, *A friendly introduction to number theory*, Prentice-Hall, 2001.
- [Tei98] Jeremy Teitelbaum, *Euclid's algorithm and the Lanczos method over finite fields*, Mathematics of Computation **67** (1998), no. 224, 1665–1678.

APPENDIX A

BACKGROUND AND NOTATION

We quote useful definitions and theorems in this section from following reference books: Artin [Art91] and Friedberg/Insel/Spence [FIS97].

Theorem A.0.1. [FIS97] p. 74

Let V and W be finite-dimensional vector spaces with ordered bases β and γ , respectively, where $\beta = \{v_1, \dots, v_n\}$ and $\gamma = \{w_1, \dots, w_m\}$. Suppose that $T : V \rightarrow W$ is linear. Then there exist unique scalars a_{ij} such that: $T(v_j) = \sum_{i=1}^m a_{ij} w_i$ for $1 \leq j \leq n$. $\square$

Definition A.0.2. In the notation of the theorem above, the $m \times n$ matrix $A = [a_{ij}]$ is known as the *matrix representation of the linear transformation T from β to γ* , or $A = [T]_{\beta}^{\gamma} = [(T(v_1))_{\gamma}, (T(v_2))_{\gamma}, \dots, (T(v_n))_{\gamma}]$. If $V = W$, and $\beta = \gamma$, we write $A = [T]_{\beta} = [(T(v_1))_{\beta}, (T(v_2))_{\beta}, \dots, (T(v_n))_{\beta}]$.

Definition A.0.3. [FIS97] p. 65

Let $T : V \rightarrow W$ be a linear transformation. Define $\text{rank}(T)$, the dimension of $R(T)$, where $R(T)$ is the range or the image of T a subspace of W .

Theorem A.0.4. [Dimension Theorem] [FIS97] p. 65

Let $T : V \rightarrow W$ be a linear transformation. If V is a finite-dimensional vector spaces then

$$\text{nullity}(T) + \text{rank}(T) = \dim(V). \quad (\text{A.1})$$

Definition A.0.5. [FIS97] p. 143

The *rank* of an m -by- n matrix A is the rank of the linear transformation L_A where $L_A : F^n \rightarrow F^m$.

Corollary A.0.6. [FIS97] p. 144 Let A be an $n \times m$ matrix. The rank of A is the dimension of its column space, and by Theorem A.0.4 we have $\text{nullity } A + \text{rank } A = n$.

Definition A.0.7. [FIS97] pp. 72-73

Let V be a vector space over a field F and let $T : V \rightarrow V$ be linear. A subspace W of V is said to be *T-invariant* if $T(x) \in W$ for every $x \in W$, i.e., $T(W) \subset W$. If W is T -invariant, we define the *restriction* of T to W to be the function $T_W : W \rightarrow W$ given by $T_W(x) = T(x)$ for all $x \in W$.

Some examples of invariant subspaces of the vector space $V = \mathbb{R}^n$ under multiplication by an $n \times n$ matrix A with real entries include:

- (1) $\{0\}$.
- (2) V .
- (3) $N(A)$, the nullspace of A .
- (4) $R(A)$, the range space or column space of A .
- (5) The eigenspace related to any eigenvalue of A .
- (6) $\text{Span}\{v_1, v_2, \dots, v_k\}$, where every v_i is an eigenvector for A .

It is trivial that $\{0\}$ and V are A -invariant. By the definition of nullspace, which is $N(A) = \{x \in V | Ax = 0\}$, if $x \in N(A)$ then $Ax = 0 \in N(A)$. We know $R(A) = \{Ax | x \in V\}$, and $R(A) \subset V$, so if $y = Ax \in R(A)$, then Ay is also in $R(A)$ since $y \in V$. The eigenspace corresponding to the eigenvalue λ of A , $E_\lambda(A)$, is also A -invariant, since $E_\lambda(A) = N(A - \lambda I)$. For the last example, it is sufficient to use the properties of the direct sum of eigenspaces of A ; see Friedberg, Insel and Spence [FIS97].

Theorem A.0.8. *[Dependency Theorem] Let S be a linearly independent subset of a finite-dimensional vector space V . For any $v \in V$ that is not in S , the set $S \cup \{v\}$ is linearly dependent if and only if $v \in \text{Span } S$.*

Proof: See Friedberg/Insel/Spence [FIS97], p. 41. □

Theorem A.0.9. *[Art91] pp. 509-515.*

Let $q = p^r$, where p is a prime number and $r \geq 1$. Then there exists a unique field of order q up to isomorphism, namely $GF_p[t]/(f(t))$, where $f(t)$ is an irreducible polynomial of degree r over GF_p .

Proof: See Artin [Art91], pp. 509-515. □