TEL AVIV UNIVERSITY אוניברסיטת תל-אביב

Raymond and Beverly Sackler Faculty of Exact Sciences
School of Computer Science

Algebraic and Algorithmic Applications of Support Theory

Thesis submitted for the degree of "Doctor of Philosophy"
by

Doron Chen

This work was carried out under the supervision of
Prof. Sivan Toledo

Submitted to the Senate of Tel-Aviv University
September 2005

To my dear parents.

Acknowledgments

I have been fortunate to collaborate with several researchers during my graduate studies: Erik G. Boman, Bruce Hendrickson, Ojas Parekh, John Gilbert, Olga Sorkine and Daniel Cohen-Or. I have enjoyed working with them and learned a great deal from them. I would like to thank them all.

I would like to thank my office-mates for their friendship and basketball games.

I would also like to thank my dear friends: Oren, Etai, Leah, Dan and Ginna.

I wish to express my deepest gratitude to my parents for their love and ongoing support. To my sister Leat and her family: Gilad, Talia and Avivush.

Most of all, I would like to thank my advisor, Sivan Toledo. One cannot hope for a better advisor, both on a personal and on a professional level. Working with Sivan has been both a wonderful learning experience and a pleasure.

Abstract

This thesis contains algebraic and algorithmic results in sparse linear algebra. All these results make use of support theory. Though the tools of support theory were originally designed in order to bound the condition numbers of preconditioned systems, we show here that these tools have a variety of other uses. The results given in this work have applications in support theory, but they can also be viewed as independent algebraic results. For instance, we show how support theory can be used in order to obtain new bounds on the 2-norm of any rectangular real matrix. We also show new applications of support theory in computer graphics and in solving singular linear systems.

This thesis consists of four main themes:

(1) **Factor Width:** We define a matrix concept called *factor width,* a measure for the combinatorial complexity of symmetric positive semidefinite (SPSD) matrices. This gives a hierarchy of matrix classes for SPSD matrices. Factor-width-2 matrices are matrices that can be represented as graphs. Factor-width-3 matrices are matrices that can be represented as hypergraphs, such that each hyperedge connects three vertices. In general, factor-width-k matrices are matrices that can be represented as hypergraphs, such that each hyperedge connects k vertices. We prove that the set of symmetric matrices with factor width at most two is exactly the class of symmetric H-matrices with non-negative diagonals. We prove bounds on the factor width, including one that is tight for factor widths up to two. These results were published in "On Factor Width and Symmetric H-matrices", *Linear Algebra and its Applications* 405: 239–248 (2005) by Erik G. Boman, Doron Chen, Ojas Parekh and Sivan Toledo.

(2) **Null Spaces of Symmetric H-matrices:** We characterize the structure of null spaces of factor-width-2 matrices. We show that the structure of the null space of a factor-width-2 matrix depends on the structure of the connected components of its underlying

graphs. Each connected component contributes at most one vector to the null space. We provide a combinatorial characterization of the rank of each connected component, and a combinatorial characterization of a null vector if one exists. For symmetric diagonally-dominant (SDD) matrices, we also present an efficient combinatorial algorithm for constructing an orthonormal basis for the null space. We show a close connection between gain graphs and symmetric H-matrices with non-negative diagonals, which extends known results regarding the connection between undirected graphs and Laplacian matrices, and between signed graphs and SDD matrices. We show how to exploit these combinatorial algorithms to reliably solve certain singular linear systems in finite-precision arithmetic. These results were published in "Combinatorial Characterization of the Null Spaces of Symmetric H-Matrices", *Linear Algebra and its Applications 392: 71–90 (2004)* by Doron Chen and Sivan Toledo.

(3) **Obtaining Bounds on the Two Norm of a Matrix:** We show how two support theory tools, the splitting lemma and the symmetric-product-support lemma, can be used to obtain new bounds on the 2-norm of a real matrix. We derive six separate algebraic bounds and analyze the combinatorial interpretations of these bounds. These results were accepted for publication in *Electronic Transactions on Numerical Analysis*, "Obtaining Bounds on the Two Norm of a Matrix from the Splitting Lemma" by Doron Chen, John R. Gilbert and Sivan Toledo.

(4) **Applications of Support Theory in Computer Graphics:** We introduce new applications of support theory in computer graphics. In most support theory applications we are given a matrix, and we then represent it as a graph in order to analyze or precondition it. In this new application, however, the process is reversed. We are given a graph which we want to compress. The compression involves the use a Laplacian-like matrix which is obtained from the connectivity graph.

We present an algebraic analysis of a mesh-compression technique called high-pass quantization. In high-pass quantization, a rectangular matrix based on the mesh topological Laplacian is applied to the vectors of the Cartesian coordinates of a polygonal mesh. The resulting vectors, called δ-coordinates, are then quantized. The applied matrix is a function of the topology of the mesh and the indices of a small set of mesh vertices (anchors), but not of the location of the vertices. An approximation of the geometry can be reconstructed from the quantized δ-coordinates and the spatial

locations of the anchors. We show how to algebraically bound the reconstruction error that this method generates. We show that the small singular value of the transformation matrix can be used to bound both the quantization error and the rounding error, which is due to the use of floating-point arithmetic. Furthermore, we prove a bound on this singular value. The bound is a function of the topology of the mesh and of the selected anchors. We also propose a new anchor-selection algorithm, inspired by this bound. We show experimentally that the method is effective and that the computed upper bound on the error is not too pessimistic. These results were accepted for publication in *ACM Transactions on Graphics*, "Algebraic Analysis of High-Pass Quantization" by Doron Chen, Daniel Cohen-Or, Olga Sorkine, and Sivan Toledo.

Contents

CHAPTER 1

Introduction

This chapter informally illustrates the main results of the thesis. The aim of this chapter is to explain what the results are, rather than to explain how we prove them. We use many small examples for clarification. Each section describes the main results in one chapter of the thesis.

1.1. Factor Width

In Chapter 2 we define a matrix concept called *factor width,* a measure for the combinatorial complexity of symmetric positive semidefinite (SPSD) matrices. This gives a hierarchy of matrix classes for SPSD matrices. Factor-width-2 matrices are matrices that can be represented as graphs. Factor-width-3 matrices are matrices that can be represented as hypergraphs, such that each hyperedge connects three vertices. In general, factor-width-k matrices are matrices that can be represented as hypergraphs, such that each hyperedge connects k vertices. This correspondence between SPSD matrices and hypergraphs extends the well-known correspondence between weighted graphs and symmetric diagonally-dominant matrices, which has been exploited in many ways in the past.

A real symmetric diagonally-dominant (SDD) matrix A can always be decomposed into a symmetric factorization $A = UU^T$, where U has at most two nonzeros per column (and in particular, all the nonzeros in a column of U have the same absolute value) [**13**]. For example, consider the matrix

$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 5 & 4 \\ 0 & 4 & 4 \end{bmatrix} .$$

Matrix A can be represented as $A = UU^T$ where

$$U = \begin{bmatrix} 1 & 0 & 1 \\ -1 & 2 & 0 \\ 0 & 2 & 0 \end{bmatrix} .$$

This decomposition is obtained in the following way:

$$\begin{aligned}
A &= \begin{bmatrix} 2 & -1 & 0 \\ -1 & 5 & 4 \\ 0 & 4 & 4 \end{bmatrix} = \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 4 \\ 0 & 4 & 4 \end{bmatrix} + \begin{bmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 4 \\ 0 & 4 & 4 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & -1 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & 4 \\ 0 & 4 & 4 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & -1 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 2 \\ 2 \end{bmatrix} \cdot \begin{bmatrix} 0 & 2 & 2 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & -1 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 2 \\ 2 \end{bmatrix} \cdot \begin{bmatrix} 0 & 2 & 2 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & -1 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 1 \\ -1 & 2 & 0 \\ 0 & 2 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 \\ -1 & 2 & 0 \\ 0 & 2 & 0 \end{bmatrix}^T .
\end{aligned}$$

Note that at each step of the decomposition we handle one off-diagonal nonzero or the diagonal entry of a strictly diagonally-dominant row.

The nonzero structure of U is that of a vertex-edge incidence matrix of the underlying graph[1] of A; the nonzero structure of A itself is that of a vertex-vertex adjacency matrix of the same graph. Figure 1.1.1 shows a matrix A, its rectangular factor U and its underlying graph.

This relationship between symmetric diagonally-dominant matrices and weighted graphs has been exploited in the past, among other things, to

- characterize graph properties, such as the size of balanced vertex separators, by eigenvalues of the corresponding matrix [**17, 31, 33, 34, 35, 47, 48, 54, 76, 77, 87**];
- to bound the smallest nonzero eigenvalue of A using structures in the graph [**30, 34, 35, 49, 50, 62, 83**];

[1]The *underlying graph* G_A of an n-by-n symmetric matrix A is a weighted graph $G_A = (V_A, E_A, w)$, where $V_A = \{1, 2, \ldots, n\}$, $E_A = \{(i,j)\colon A_{ij} \neq 0\}$, and the weight of an edge (i,j) is $w(i,j) = -A_{ij}$.

$$A = \begin{bmatrix} 11 & -2 & -5 & & -4 & & & \\ -2 & 13 & -5 & & & -6 & & \\ -5 & -5 & 17 & -6 & -1 & & & \\ & & -6 & 8 & & -2 & & \\ -4 & & -1 & & 12 & -3 & -4 & \\ & -6 & & -2 & -3 & 16 & -5 & \\ & & & & -4 & -5 & 15 & -6 \\ & & & & & & -6 & 6 \end{bmatrix}$$

$$U = \begin{bmatrix} \sqrt{2} & \sqrt{5} & \sqrt{4} & & & & & & & & & \\ -\sqrt{2} & & & \sqrt{5} & \sqrt{6} & & & & & & & \\ & -\sqrt{5} & & -\sqrt{5} & & \sqrt{6} & \sqrt{1} & & & & & \\ & & -\sqrt{4} & & & -\sqrt{6} & & \sqrt{2} & & & & \\ & & & & & & -\sqrt{1} & & \sqrt{3} & \sqrt{4} & & \\ & & & & -\sqrt{6} & & & -\sqrt{2} & -\sqrt{3} & & \sqrt{5} & \\ & & & & & & & & & -\sqrt{4} & -\sqrt{5} & \sqrt{6} \\ & & & & & & & & & & & -\sqrt{6} \end{bmatrix}$$

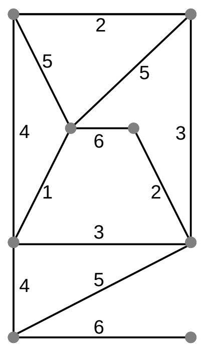


FIGURE 1.1.1. An SDD matrix A (top), its rectangular factor U (middle) and the graph G_A of A (bottom). The vertices are ordered top to bottom, left to right.

- to bound the generalized spectral condition number of a matrix pencil in terms of embeddings of the corresponding graphs [**9, 16, 42**];
- to design preconditioners by sparsifying the corresponding graph [**9, 13, 93**] or by constructing a related graph [**42**];
- to obtain combinatorial graph algorithms for computing the null space of symmetric diagonally-dominant matrices [**22**].

We define the factor width of a matrix A as the smallest integer k such that A has a symmetric factorization $A = UU^T$ with at most k nonzeros in each column of U. We argue that the factor width is a good measure of the "combinatorial complexity" of a matrix. A symmetric diagonally-dominant matrix is combinatorially simple in the sense that it has an incidence factor U that corresponds to a weighted undirected graph. A factor-width-3 matrix does not have such a factor, but it does have an incidence factor that corresponds to a hypergraph with at most 3 vertices per edge.

Clearly, not all factor-width-2 matrices are diagonally dominant. For instance

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$$

is not diagonally dominant.

Also, it is clear that not all symmetric positive semi-definite matrices have factor width 2. Let

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} .$$

Matrix A can be represented as

$$A = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} ,$$

so its factor width is at most 3. Also, A cannot have a factor-width-2 representation: matrix A has rank-1, so any representation of A as $A = UU^T$ must have the property that the columns of U are identical up to a multiplicative constant. Therefore, without loss of generality U must have a single column, and it is trivial to show that this column is the vector of ones.

The main results of Chapter 2 are:

THEOREM. *A matrix has factor width at most two if and only if it is a symmetric H^+-matrix. We use H^+ to denote H-matrices that have non-negative diagonal.*

DEFINITION. Let dn(A) denote the symmetric scaling of A so that the diagonal elements of the scaled matrix are all 1's (except for diagonal elements in zero rows of A, which remain zero).

THEOREM. *For any SPSD matrix A, the factor width of A is bounded from below by $\lceil \| dn(A) \|_2 \rceil$.*

We also proved a tighter lower bound (illustrated in figure 1.1.2):

THEOREM. *For any SPSD matrix A, the factor width of A is bounded from below by $\lceil \| dn(|A|) \|_2 \rceil$. $|A|$ denotes the matrix whose (i, j) entry is $|A_{ij}|$.*

$$A = \begin{bmatrix} 5 & -3 \\ -3 & 2 \end{bmatrix} \qquad |A| = \begin{bmatrix} 5 & 3 \\ 3 & 2 \end{bmatrix}$$

$$\operatorname{dn}(|A|) = \begin{bmatrix} 1 & \frac{3}{\sqrt{10}} \\ \frac{3}{\sqrt{10}} & 1 \end{bmatrix} \qquad \lceil \|\operatorname{dn}(|A|)\|_2 \rceil = \left\lceil \left\| \begin{bmatrix} 1 & \frac{3}{\sqrt{10}} \\ \frac{3}{\sqrt{10}} & 1 \end{bmatrix} \right\| \right\rceil = \lceil 1.9487 \rceil = 2$$

FIGURE 1.1.2. An example of a lower bound on the width factor.

THEOREM. *Matrix A has factor-width at most 2 if and only if it is symmetric with non-negative diagonals, and satisfies $\|dn(|A|)\|_2 \leq 2$.*

The beauty of these bounds lies in that they link a discrete combinatorial measurement, namely the factor width of a matrix, with the norm of a matrix, which is a continuous measurement. The results in Chapter 2 were published in "On Factor Width and Symmetric H-matrices", *Linear Algebra and its Applications* 405: 239–248 (2005) by Erik G. Boman, Doron Chen, Ojas Parekh and Sivan Toledo.

1.2. Null Spaces of Symmetric H-matrices

In Chapter 3 we characterize the structure of a basis for the null spaces of factor-width-2 matrices (which we have shown to be the class of symmetric H-matrices with non-negative diagonals, or symmetric H^+-matrices). The characterization relies on combinatorial properties of a graphs associated with these factor-width-2 matrices. We present combinatorial algorithms for determining the rank and constructing an orthonormal basis for the null space of a factor-width-2 matrix, given its factor-width-2 factorization.

We investigate the close connection between symmetric H-matrices with non-negative diagonals and gain graphs [**79, 96**] (previously called *voltage graphs* [**43, 44**]). A gain graph is an undirected graph such that each edge e can be viewed as two edges, one in each direction, with a different weight in each direction. These weights are called *gain*, and they have the following property: Let e be an edge connecting vertex i and vertex j. Then e has two gains: a gain g from vertex i to vertex j, and a gain $1/g$ from vertex j to vertex i.

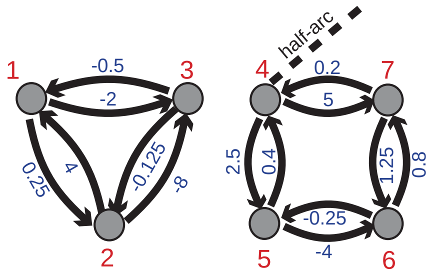
The class of factor-width-2 matrices includes all the matrices that can be factored into $A = UU^T$, such that U has at most two nonzeros per column. The graph associated with A is constructed as follows: we represent each row of A as a vertex; each column u of U is represented by an edge. If u

has a two nonzeros, α in the i-th position and β in the j-th position, then we represent it as an edge connecting vertices i and j; the gain of this edge is $-\beta/\alpha$ between i and j, and $-\alpha/\beta$ between j and i. For columns with a single nonzero, we attach a *half-arc*, which is an edge with a single endpoint and no gain; if u has a single nonzero in position i, then we represent it as a half arc incident to vertex i.

As an example, consider $A = UU^T$, where

$$U = \begin{bmatrix} 4 & & 1 & & & & & \\ -1 & 1 & & & & & & \\ & 8 & 2 & & & & & \\ & & & 2 & & & -1 & -3 \\ & & & -5 & 2 & & & \\ & & & & 8 & 5 & & \\ & & & & & -4 & 5 & \end{bmatrix}.$$

The graph corresponding to this matrix is



Note that the nonzero structure of U can be viewed as the incidence matrix of the underlying graph of A.

In Chapter 3 we show that the structure of the null space of a symmetric H-matrix depends on the structure of the connected components of its underlying graph. Each connected component contributes at most one vector to the null space.

Whether or not a connected component contributes a vector to the null space, depends on its *balance*. In order to define the balance of a connected component, we need the following definitions: The *gain of a directed path* or *cycle* is the product of the gains of the edges along the path or cycle. A cycle is *balanced* if its gain is exactly $+1$. A connected component is called *balanced* if it contains no half arcs and no unbalanced cycles. We have shown that each balanced component contributes one vector to the null space, whereas each unbalanced component has full rank. In our example, the left connected component contains one cycle which is balanced and contains no half arcs. As a result, the left component is balanced and contributes one vector to the null space. The right component is unbalanced and therefore has full-rank.

We provide a combinatorial characterization of the null vectors of each balanced connected component. It turns out that in order to construct a null vector of a balanced component, we need to choose some arbitrary vertex in the component. We call that vertex a *root*. Let v be the null vector associated with the component. Then $v_j = 0$ if and only if vertex j is not in the component. For every vertex j in the connected component, v_j is the gain of a path between j and the root. This gain is well-defined, because in a balanced connected component it does not matter which path we choose between vertex j and the root; all paths must have the same gain.

In our example, let us choose vertex 2 to be the root of the left connected component. Then this component contributes the following vector to the null space:

$$\begin{bmatrix} 0.25 \\ 1 \\ -0.125 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

The number of cycles in a graph is typically great; nevertheless, we present an efficient combinatorial algorithm for determining the rank and constructing an orthonormal basis for the null space of an factor-width-2 matrix, given its factor-width-2 factorization. We show how to exploit this combinatorial algorithm to reliably solve certain singular linear systems in finite-precision arithmetic. The results in this chapter 3 were published in "Combinatorial Characterization of the Null Spaces of Symmetric H-Matrices", *Linear Algebra and its Applications 392: 71–90 (2004)* by Doron Chen and Sivan Toledo.

1.3. Obtaining Bounds on the Two Norm of a Matrix

In Chapter 4, we prove several new bounds on the 2-norm of a general rectangular real matrix. The bounds exploit the sparsity of W, and are often tighter than other well-known bounds such as the Frobenius norm. Let W be a k-by-m real matrix. We show that:

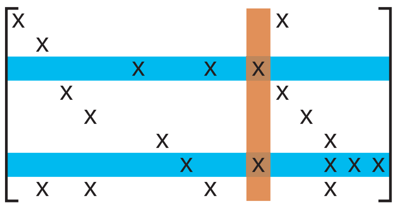


FIGURE 1.3.1. An illustration of the bound $\|W\|_2^2 \leq \max_j \sum_{i:\, W_{i,j}\neq 0} \|W_{i,:}\|_2^2$. The X-s in this figure denote nonzeros of W. The bound states that $\|W\|_2^2$ is bounded by the maximum, over all the columns j of W, of the sum of the squares of the 2-norms of all the rows in W which contain a nonzero in the j-th column.

$$\begin{aligned}
\|W\|_2^2 &\leq \max_j \sum_{i:\, W_{i,j}\neq 0} \|W_{i,:}\|_2^2 = \max_j \sum_{i:\, W_{i,j}\neq 0} \sum_{c=1}^{m} W_{i,c}^2 \\
\|W\|_2^2 &\leq \max_i \sum_{j:\, W_{i,j}\neq 0} \|W_{:,j}\|_2^2 = \max_i \sum_{j:\, W_{i,j}\neq 0} \sum_{r=1}^{k} W_{r,j}^2 \\
\|W\|_2^2 &\leq \max_j \sum_{i:\, W_{i,j}\neq 0} |W_{i,j}| \cdot \left(\sum_{c=1}^{m} |W_{i,c}| \right) \\
\|W\|_2^2 &\leq \max_i \sum_{j:\, W_{i,j}\neq 0} |W_{i,j}| \cdot \left(\sum_{r=1}^{k} |W_{r,j}| \right) \\
\|W\|_2^2 &\leq \left\|WW^T\right\|_1 \\
\|W\|_2^2 &\leq \left\|W^TW\right\|_1
\end{aligned}$$

The first bound states that the 2-norm of W squared, is bounded by the maximum, over all the columns j of W, of the sum of the squares of the 2-norms of all the rows in W which contain a nonzero in the j-th column. Figure 1.3.1 illustrates this bound. This bound is clearly no worse (and often much better) than the Frobenius bound which is the sum of the square of the 2-norms of *all* the rows.

Although these bounds are general and apply to any matrix, the motivation for the analysis that yielded them came from support theory. The proofs of these bounds also use support-theory tools.

How are are norm bounds connected with support theory? Boman and Hendrickson [**16**] and Boman, Chen, Hendrickson and Toledo [**13**] showed that when an SDD matrix A is preconditioned by another SDD matrix B, the convergence of the preconditioned iterative method can be bounded by the product of the 2-norms of two matrices, W and Z. These matrices are not uniquely defined, but an embedding of the edges of the graph of A into paths in the graph of B defines a W, and an embedding of the edges of G_B in paths in G_A defines a Z. When W is defined by an embedding, well-known bounds on $\|W\|_2$ can be interpreted as combinatorial measures of the quality of the embedding. For example, the Frobenius norm, which bounds the 2-norm, is proportional to the average stretch of edges in the embedding, for an appropriate definition of stretch [**14**].

When W is defined by an embedding, our new bounds on the 2-norm can be interpreted as new combinatorial measures of the embedding. In fact, the analysis that led to these new norm bounds was motivated by a new combinatorial measure of embeddings that was used by Spielman and Teng to bound convergence rates of a preconditioned iterative method.

The results in chapter 4 were accepted for publication in *Electronic Transactions on Numerical Analysis*, "Obtaining Bounds on the Two Norm of a Matrix from the Splitting Lemma" by Doron Chen, John R. Gilbert and Sivan Toledo.

1.4. Applications of Support Theory in Computer Graphics

In Chapter 5 we use support theory tools to analyze a mesh-compression technique called *high-pass quantization* [**85**]. In computer graphics, objects are often represented as a three-dimensional mesh: a set of triangles that share edges and vertices. Except on the boundary of the mesh, each edge is shared by two triangles. The mesh is represented by a finite set of vertices, a finite set of triangles, and geometrical information. The geometrical information usually consists of Cartesian coordinates for each vertex. Abstractly, we can view the mesh as a graph or hypergraph and several vectors whose elements are associated with vertices, an x vector, a y vector, and a z vector.

Mesh compression involves two problems that are usually solved separately: the mesh *connectivity* encoding and the *geometry* encoding. While state-of-the-art connectivity encoding techniques are extremely effective [**1, 51, 65, 91**], compressing the geometry remains a challenge. The raw geometry data usually comes in high-precision floating-point representation. Such data cannot be significantly compressed using standard compression techniques; therefore, most geometry encoding schemes involve quantization, which introduces errors and causes certain loss of data.

The quantization can be applied directly to the Cartesian coordinates, or to some invertible transformation of them. In *high-pass mesh quantization*, quantization is applied after the coordinates are transformed using a transformation that belongs to a particular class of linear operators [**85**].

In Chapter 5 we present an algebraic analysis of the high-pass quantization technique. We show how to algebraically bound the reconstruction error that this method generates. We show that the small singular value of the transformation matrix can be used to bound both the quantization error and the rounding error, which is due to the use of floating-point arithmetic. Furthermore, we use support theory tools to prove a bound on this singular value. We show experimentally that the method is effective and that the computed upper bound on the error is not too pessimistic. The results in chapter 5 were accepted for publication in *ACM Transactions on Graphics*, "Algebraic Analysis of High-Pass Quantization" by Doron Chen, Daniel Cohen-Or, Olga Sorkine, and Sivan Toledo.

We now use a simple two-dimensional version of high-pass mesh quantization to illustrate the method and our analysis technique. Let m be a closed polygon in the plane. When viewed as a mesh, the edges of the polygon are faces of the mesh. Since the polygon is closed, the graph of the mesh is a simple cycle. Each mesh point is associated with x and y values. Applying quantization to the x and y values directly produces high-frequency errors that can be visually displeasing, as shown in Figure 1.4.1.

To alleviate this, high-pass quantization first transforms the coordinates using a Laplacian-like matrix. The transformed coordinates are called δ-coordinates. The transformation works as follows. The δ coordinates of mesh point i are

$$\delta_i^{(x)} = 2x_i - x_{i-1} - x_{i+1} = \text{degree}(i)\left(x_i - \frac{\sum_{j \text{ is a neighbor of } i} x_j}{\text{degree}(i)}\right)$$

and similarly for $\delta_i^{(y)}$. The expression within the parenthesis is the distance from the x coordinate of i to the average of its neighbors. When m approximates a smooth curve c near point i, the average of the neighbors tends to be a good predictor for x_i, so the δ coordinate tends to be smaller than the Cartesian coordinates. In matrix form, the transformation is

$$\delta^{(x)} = \begin{bmatrix} 2 & -1 & & & & -1 \\ -1 & 2 & -1 & & & \\ & -1 & 2 & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & 2 & -1 \\ -1 & & & & -1 & 2 \end{bmatrix} x = Lx \ .$$

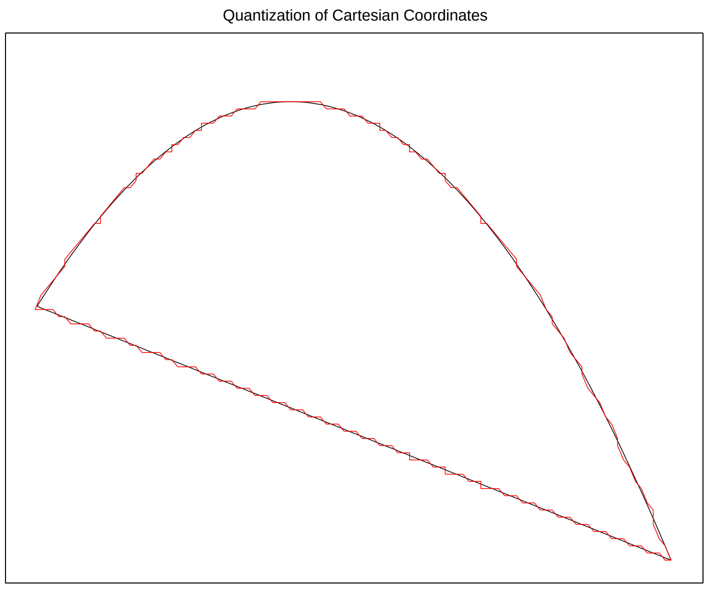


FIGURE 1.4.1. A polygon (black) representing a mesh, and the polygon (red) that results from quantizing the x and y coordinates of the polygon using 64 values.

The transformation matrix L is singular, because its row sums are exactly 0. We invert the transformation by specifying one more constraint, say the original Cartesian coordinate of one mesh point, say x_1. We call such a point an *anchor*.

With a single anchor, the transformation is invertible, but ill conditioned: the smallest singular value of the transformation is small, which implies that the inverse has a large norm. This causes large reconstruction errors, as shown in the example in Figure 1.4.2. To improve the conditioning of the transformation, we can add anchors. The δ-coordinates and the Cartesian coordinates of the anchors define the original Cartesian coordinates via the implicit equations

$$\begin{aligned} Lx &= \delta^{(x)} \\ x_{k_1} &= b_{k_1} \\ &\vdots \\ x_{k_\ell} &= b_{k_\ell} \,, \end{aligned}$$

where $k_1, \ldots, k_\ell$ are the indices of the anchors. However, if we try to reconstruct an approximation for x given *quantized* δ coordinates, we cannot use this system of equations, because in general it is inconsistent. Instead, high-pass mesh quantization reconstructs approximate coordinates by solving this system in the least squares sense.

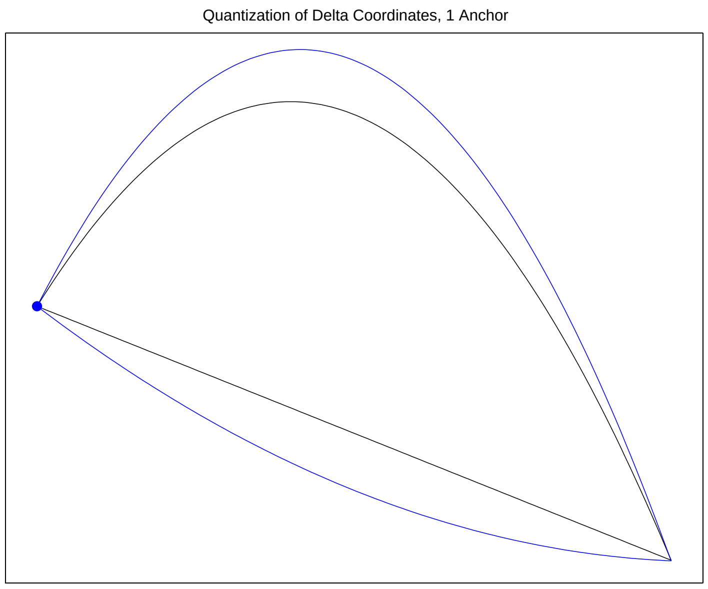


FIGURE 1.4.2. A polygon (black) representing a mesh, and the reconstruction of it (blue) from a quantization of the δ-coordinates Lx and Ly, again using 64 values. The reconstruction uses one anchor point, marked by a circle.

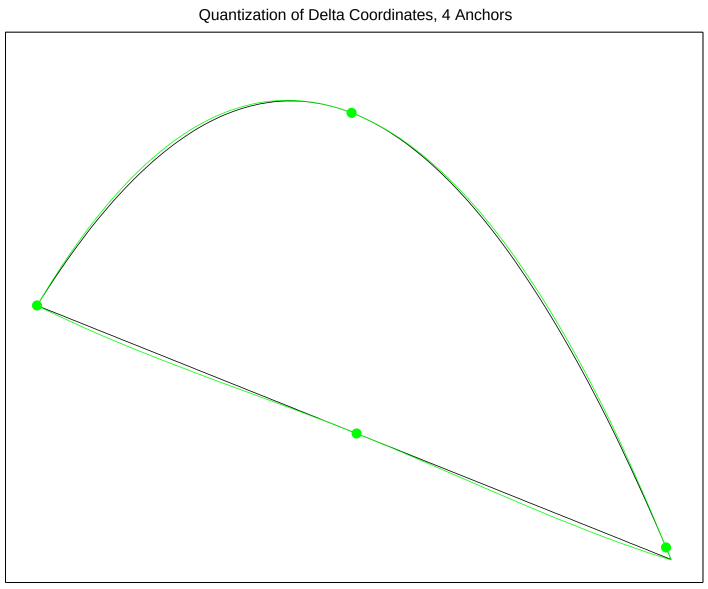


FIGURE 1.4.3. A polygon (black) representing a mesh, and the reconstruction of it (blue) from a quantization of the δ-coordinates. This time, the reconstruction uses 4 anchor points, marked by circles.

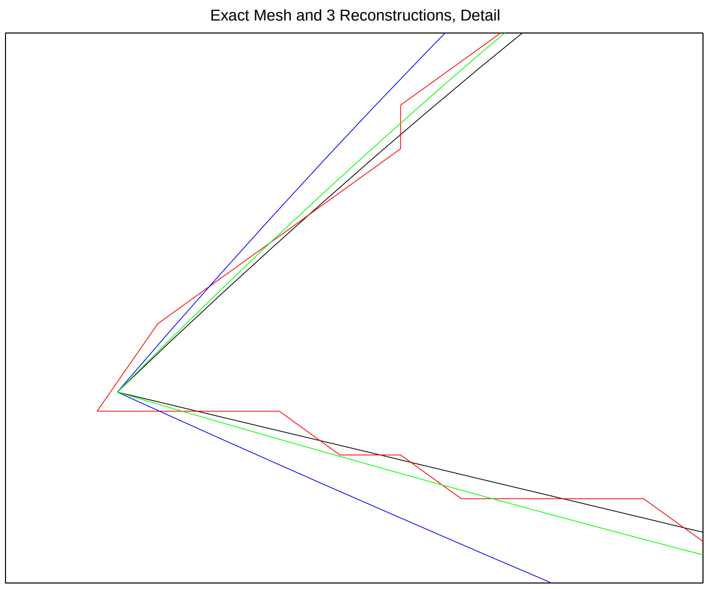


FIGURE 1.4.4. The original polygon (black) and the three reconstructions from quantized coordinates. The figure shows only the leftmost part of the four polygons, to show the qualitative difference in the errors.

We show in Chapter 5 that the quantization errors and the rounding errors can be bounded by a function of the smallest singular value of the coefficient matrix

$$\tilde{L} = \begin{bmatrix} 2 & -1 & & & & -1 \\ -1 & 2 & -1 & & & \\ & -1 & 2 & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & 2 & -1 \\ -1 & & & & -1 & 2 \\ 1 & & & & & \\ & & 1 & & & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ & & & & 1 & \end{bmatrix}$$

of the least-squares problem. Furthermore, we also show how to bound this singular value using combinatorial support arguments. Figure 1.4.3 uses the example polygon to show the reconstruction with 4 anchors. Figure 1.4.4 zooms in on a detail of the original polygon, showing the three reconstructions.

CHAPTER 2

On Factor Width and Symmetric H-matrices[1]

2.1. Introduction

Symmetric positive definite and semidefinite (SPD and SPSD, respectively) matrices arise frequently in applications and have been studied by many authors [**8, 60**]. For instance, it is well known that a Cholesky decomposition $A = LL^T$, where L is lower triangular, exists for any SPD matrix A. In this chapter we characterize SPSD matrices in terms of rectangular factorizations of the type $A = VV^T$, where V is typically sparse and may have more columns than rows.

We restrict our attention to real matrices in this chapter. In Section 2.2 we define the factor width of a symmetric matrix and show some basic properties. In Section 2.3 we show our main result, that factor-width-2 matrices are precisely H^+ matrices. We review a couple of known properties of H-matrices in the process. In Section 2.4 we prove bounds on the factor width, and show that a lower bound is exact for factor widths one and two. Finally, in Section 2.6 we pose several open questions.

2.2. The Factor Width of a Symmetric Matrix

DEFINITION 2.2.1. The factor width of a real symmetric matrix A is the smallest integer k such that there exists a real (rectangular) matrix V where $A = VV^T$ and each column of V contains at most k non-zeros.

For example, let

$$A = \begin{pmatrix} 3 & 1 & -1 \\ 1 & 2 & -2 \\ -1 & -2 & 5 \end{pmatrix}, \text{ and let } V = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & -2 & -1 \end{pmatrix}.$$

Then A has factor width at most two because $A = VV^T$. It is easy to see that a matrix has factor width one if and only if it is diagonal and non-negative; hence, the factor width of A is two. The factor width is independent of the ordering of the matrix since $PAP^T = (PV)(PV)^T$ has the same factor width as $A = VV^T$ for any permutation matrix P.

[1]The results in this chapter were published in "On Factor Width and Symmetric H-matrices", *Linear Algebra and its Applications* 405: 239–248 (2005) by Erik G. Boman, Doron Chen, Ojas Parekh and Sivan Toledo.

It follows from well-known properties of diagonally dominant matrices [**5**] that symmetric diagonally dominant matrices with non-negative diagonal have factor width two, which we also prove below. Recall that a real matrix A is diagonally dominant if $|a_{ii}| \geq \sum_{j \neq i} |a_{ij}|$ for all i.

PROPOSITION 2.2.2. *If A is SPSD and diagonally dominant then A has factor width at most two.*

PROOF. Let $P = \{(i,j) | i < j, a_{ij} > 0\}$ and $N = \{(i,j) | i < j, a_{ij} < 0\}$, and let e_i denote the ith unit vector. Then we can write A as a sum of rank-1 matrices,

$$\begin{aligned} A &= \sum_{i=1}^{n} \left(a_{ii} - \sum_{j \neq i} |a_{ij}| \right) e_i e_i^T \\ &\quad + \sum_{(i,j) \in P} a_{ij} (e_i + e_j)(e_i + e_j)^T \\ &\quad + \sum_{(i,j) \in N} (-a_{ij})(e_i - e_j)(e_i - e_j)^T \,. \end{aligned}$$

For diagonally dominant matrices all the coefficients are non-negative, and one can readily construct a V such that $A = VV^T$ from the expression above where each column of V is of one of the types $\sqrt{a_{ii} - \sum_{j \neq i} |a_{ij}|}\, e_i$, $\sqrt{a_{ij}}\,(e_i + e_j)$, or $\sqrt{-a_{ij}}\,(e_i - e_j)$. $\square$

Note that not all factor-width-two matrices are diagonally dominant, as the matrix A in the beginning of this section shows. Any SPSD matrix of order n has factor width at most n. A question arises: Are there matrices of factor width k for all $k \leq n$? The answer is yes.

PROPOSITION 2.2.3. *For any positive $k \leq n$, there exist matrices of order n with factor width k.*

PROOF. Let $v_k = (1, 1, \ldots, 1, 0, \ldots, 0)^T$, where there are k ones. Let $A = v_k v_k^T$. Clearly, the factor width of A is at most k. Since A has rank one then $A = v_k v_k^T$ is the unique symmetric rank-one factorization, and there cannot be any other factorization $A = \bar{V}\bar{V}^T$ with fewer nonzeros. $\square$

We remark that the lemma above holds even if we restrict our attention to full-rank matrices. A simple example is $A = v_k v_k^T + \epsilon I$ for sufficiently small ϵ.

In conclusion, the concept of factor width defines a family of matrix classes. Let $FW(k)$ denote the set of matrices with factor width k or less. Then $FW(1) \subset FW(2) \subset \cdots$. It is easy to verify that $FW(k)$ is a pointed convex cone (see for instance, [**10**, Chapter 2]) for any k and $FW(n)$ is precisely the cone of SPSD matrices of order n.

2.3. Factor-Width-2 Matrices are H-Matrices

The importance of our study of the class of factor-width-2 matrices stems from the fact, which we prove in this section, that this class is exactly the class of H^+ matrices (defined later in this section), which occur frequently in engineering and scientific computation [**8, 70**].

The definition of H-matrices relies on M-matrices. In this thesis, we allow both M- and H- matrices to be singular, which is a bit unusual but convenient for us. Following [**8**], we have:

Definition 2.3.1. A real matrix A is an M-matrix if it is of the form $A = sI - B$, where $B \geq 0$ and $s \geq \rho(B)$, where ρ denotes the spectral radius.

For symmetric matrices there is a simpler characterization.

Lemma 2.3.2. *A real symmetric matrix A is an M-matrix if and only if $a_{ij} \leq 0$ for all $i \neq j$ and A is positive semidefinite.*

Definition 2.3.3. A matrix A is defined to be an *H-matrix* if $M(A)$ is an M-matrix, where the *comparison matrix* $M(A)$ of a matrix A is defined by

$$(M(A))_{ij} = \begin{cases} |a_{ij}|, & i = j, \\ -|a_{ij}|, & i \neq j. \end{cases}$$

We use H^+ to denote H-matrices that have non-negative diagonal. A useful characteristic of nonsingular H-matrices (see for instance, [**4**, lemma 6.4]) is that they are generalized strictly diagonally dominant, defined as follows:

Definition 2.3.4. A square matrix A is *generalized (weakly) diagonally dominant* if there exists a positive vector $y > 0$ such that for every row i,

$$|a_{ii}|\, y_i \geq \sum_{j \neq i} |a_{ij}|\, y_j\, .$$

If strict inequality holds, we say A is strictly generalized diagonally dominant.

The problem of finding such a vector y is equivalent to the problem of finding a positive diagonal matrix D such that AD (or equivalently, DAD) is diagonally dominant. This problem has been studied in [**69, 70**]; in general y may be found by solving a linear feasibility problem, but potentially faster iterative algorithms were proposed in the papers mentioned.

Theorem 2.3.5. *A symmetric matrix A is an H-matrix if and only if A is generalized (weakly) diagonally dominant.*

This is a well-known equivalence (see, e.g., [**8**] for a proof for the non-singular case). Now we are ready to prove our main result.

THEOREM 2.3.6. *A matrix has factor width at most two if and only if it is a symmetric H^+-matrix.*

PROOF. ($\Leftarrow$) Suppose that A is a symmetric H^+-matrix. Then A is generalized diagonally dominant, and hence there is a positive diagonal matrix D such that $\tilde{A} = DAD$ and $\tilde{A}$ is diagonally dominant. We know that diagonally dominant matrices have factor-width at most 2 by Proposition 2.2.2. Hence $\tilde{A} = VV^T$ for some V with at most two non-zeros per column. But $A = D^{-1}\tilde{A}D^{-1} = (D^{-1}V)(D^{-1}V)^T$, so A also has factor-width 2. This concludes the first part of the proof.

($\Rightarrow$) Suppose A has factor width two or less. A symmetric matrix A is an H-matrix if and only if its comparison matrix $M(A)$ is an M-matrix. Given a factor-width-two factorization $A = VV^T$, we can obtain a width-two factorization of $M(A)$ by simply flipping the sign of one nonzero in each column of V that contains two nonzeros with the same sign. By this factorization, $M(A)$ is positive semidefinite. Because $M(A)$ is a comparison matrix, it has nonnegative diagonals and nonpositive off-diagonals. Therefore, $M(A)$ satisfies the conditions of Lemma 2.3.2, so it is an M-matrix and hence A is an H-matrix. Since A is also SPSD, A must be in H^+. $\square$

One consequence of this theorem is that for any FW(2) matrix $A = VV^T$, there exists a positive diagonal D such that $A = (DU)(DU)^T$, where U has $\leq$ two nonzeros per column and entries of unit magnitude. However, this does not imply that $V = DU$.

2.4. Bounding the Factor Width

We do not know if the factor width of a given matrix can be efficiently computed, except in special cases. From our characterizations, it follows that $FW(k)$ matrices can be recognized in linear time for $k = 1$ and in polynomial time for $k = 2$, but already recognition for $k = 3$ is of unknown complexity and may be NP-hard.

In this section we derive several bounds that can be used to efficiently estimate the factor width of a matrix.

One upper bound on the factor width is easy to obtain: the largest number of nonzeros in a column of a Cholesky factor of PAP^T for some permutation matrix P. Many sparse matrices have a sparse Cholesky factor, and effective algorithms exist to find a permutation P that leads to a sparse factor. We note, however, that this bound may be very loose. For example, all the Cholesky factors of symmetric permutations of the Laplacian of the complete graph K_n have n nonzeros in their first column, giving a trivial upper bound of n, even though the Laplacian matrix actually has factor width 2.

The lower bounds that we present relate the factor width of a matrix A to the 2-norm of a matrix derived from A. The derivations are computationally trivial. One of the bounds is tight for matrices with factor widths one or two.

We use two tools to derive from A a matrix whose 2-norm lower bounds the factor width of A. The first tool is *diagonal normalization*, or symmetric diagonal scaling. The factor width of A is invariant under symmetric diagonal scalings of the form DAD, where D is diagonal, but the norm is not. If, however, we always symmetrically scale A so that the diagonal elements of DAD are all 1's (except for diagonal elements in zero rows of A, which remain zero in DAD), then $\|DAD\|_2$ bounds from below the factor width of A. The second tool is perhaps more surprising. We show that if we also replace the elements of A by their absolute values, we get a tighter lower bound.

DEFINITION 2.4.1. Let A be an SPSD matrix. Let D_A be the diagonal matrix whose diagonal elements are those of A, and let D_A^+ be the Moore-Penrose pseudo-inverse of D_A, that is, $(D_A^+)_{ii} = 1/(D_A)_{ii}$ for all i where $(D_A)_{ii} \neq 0$. The *diagonal normalization* $\mathrm{dn}(A)$ is the matrix

$$\mathrm{dn}(A) = \left(D_A^+\right)^{1/2} A \left(D_A^+\right)^{1/2} .$$

Our lower bounds depend on the following lemma, which provides a sufficient condition for a real function s of a matrix to be a lower bound on factor width.

LEMMA 2.4.2. *Let s be a function which assigns a real value to an SPSD matrix. Let s satisfy:*

1) $s(uu^T) \leq k$ for any column vector u with k nonzeros.
2) $s(A+B) \leq \max(s(A), s(B))$.

Then for any SPSD matrix A, the factor width of A is at least $\lceil s(A) \rceil$.

PROOF. We prove the theorem by first showing that if the factor-width of A is bounded by k, then $s(A) \leq k$.

Let A be a matrix in $FW(k)$ (A has factor width at most k). Let $A = UU^T$ be a factor-width-k representation of A. The number of nonzeros in a column u of U, which we denote by $\mathrm{nnz}(k)$, is at most k. For notational convenience, let $u \in U$ mean "u is a column of U".

$$\begin{aligned} s(A) &= s(UU^T) \\ &= s\left(\sum_{u \in U} uu^T\right) \\ &\leq \max_{u \in U}\left(s(uu^T)\right) \\ &\leq \max_{u \in U}\left(\operatorname{nnz}(u)\right) \\ &\leq k\ . \end{aligned}$$

We have shown that if the factor-width of A is at most k, then $s(A) \leq k$. Therefore, if $s(A) > k$ then the factor-width of A is larger than k. Thus, the factor width of A is greater or equal to $\lceil s(A) \rceil$. □

2.4.1. The Diagonal Normalization Bound. We now show that the factor-width of an SPSD matrix A is bounded from below by $\lceil \|\operatorname{dn}(A)\|_2 \rceil$.

THEOREM 2.4.3. *For any SPSD matrix A, the factor width of A is bounded from below by*$\lceil \|dn(A)\| \rceil$.

In our proof we will use the two results below, which we state without proofs since they both can be easily verified.

LEMMA 2.4.4. *Suppose a, b, c, d are non-negative and $c > 0, d > 0$. Then*

$$\frac{a+b}{c+d} \leq \max\left(\frac{a}{c}, \frac{b}{d}\right)$$

LEMMA 2.4.5. *Let A be SPSD. Then*

$$\|dn(A)\| = \lambda_{\max}(A, D_A) = \max_x \frac{x^T A x}{x^T D_A x},$$

where $\lambda(A, B)$ denotes a generalized eigenvalue.

We are now in position to prove Theorem 2.4.3.

PROOF. We define the function s_1 to be $s_1(A) = \|\operatorname{dn}(A)\|_2$ and show that s_1 satisfies the conditions of Lemma 2.4.2. We begin with condition 1, and show that for any vector u with k nonzeros, $s_1(uu^T)$ is exactly k.

Let u be a column vector with k nonzero entries. If $u = 0$, then $s_1(uu^T) = 0 = k$. Otherwise, uu^T is a rank-1 matrix. The matrix

$$\operatorname{dn}(uu^T) = \left(D^+_{uu^T}\right)^{1/2} uu^T \left(D^+_{uu^T}\right)^{1/2} = \left[\left(D^+_{uu^T}\right)^{1/2} u\right]\left[\left(D^+_{uu^T}\right)^{1/2} u\right]^T$$

also has rank-1, because $(D_{uu^T})_{ii} = u_i^2$. The norm of $\mathrm{dn}(uu^T)$ is the only nonzero eigenvalue of $\left(D^+_{uu^T}\right)^{1/2} uu^T \left(D^+_{uu^T}\right)^{1/2}$. Let v be the sign vector of u,

$$v_i = \begin{cases} 1 & u_i > 0, \\ -1 & u_i < 0, \\ 0 & u_i = 0. \end{cases}$$

We now show that v is an eigenvector corresponding to the eigenvalue k. We have

$$\left(D^+_{uu^T}\right)_{ii} = \begin{cases} u_i^{-2} & u_i \neq 0, \\ 0 & u_i = 0, \end{cases}$$

so

$$\left(D^+_{uu^T}\right)^{1/2} u = v \; .$$

Therefore, $\mathrm{dn}(uu^T) = vv^T$, so $\mathrm{dn}(uu^T)v = (vv^T)v = v(v^Tv) = vk = kv$.

All that remains is to prove that $s_1(A+B) \leq \max(s_1(A), s_1(B))$.

$$\begin{aligned} s_1(A+B) &= \|\mathrm{dn}(A+B)\|_2 \\ &= \max_x \frac{x^T(A+B)x}{x^T(D_A+D_B)x} \\ &= \max_x \frac{x^TAx + x^TBx}{x^TD_Ax + x^TD_Bx} \\ &\leq \max_x \max\left(\frac{x^TAx}{x^TD_Ax}, \frac{x^TBx}{x^TD_Bx}\right) \\ &\leq \max\left(\max_y \frac{y^TAy}{y^TD_Ay}, \max_z \frac{z^TBz}{z^TD_Bz}\right) \\ &= \max\left(\|\mathrm{dn}(A)\|_2, \|\mathrm{dn}(B)\|_2\right) \\ &= \max(s_1(A), s_1(B)), \end{aligned}$$

where we used Lemmas (2.4.4–2.4.5). □

2.4.2. A Tighter Lower Bound. The lower bound can be made tighter. Let $|A|$ denote the matrix whose i,j entry is $|a_{ij}|$.

THEOREM 2.4.6. *For any SPSD matrix A, the factor width of A is bounded from below by $\lceil \|dn(|A|)\|_2 \rceil$.*

PROOF. Let $s_2(A) = \lceil \|\mathrm{dn}(|A|)\|_2 \rceil$. One can show that s_2 satisfies the first condition in Lemma 2.4.2 in the same way as for s_1 (cf. proof of Theorem 2.4.3). For the second condition, we only need to prove that

$$\max_x \frac{x^T(|A+B|)x}{x^T(D_A+D_B)x} \leq \max_y \frac{y^T(|A|+|B|)y}{y^T(D_A+D_B)y}$$

because the rest follows from the previous proof. Without loss of generality, we can assume that the vector x which maximizes

$$\max_x \frac{x^T(|A+B|)x}{x^T(D_A+D_B)x}$$

is non-negative. This is due to the fact that for each vector x,

$$\frac{x^T(|A+B|)x}{x^T(D_A+D_B)x} \leq \frac{|x|^T(|A+B|)\,|x|}{|x|^T(D_A+D_B)\,|x|} .$$

Similarly, we can assume that the vector y which maximizes

$$\max_y \frac{y^T(|A|+|B|)y}{y^T(D_A+D_B)y}$$

is also non-negative.

Furthermore, for each $x \geq 0$,

$$\frac{x^T(|A+B|)x}{x^T(D_A+D_B)x} \leq \frac{x^T(|A|+|B|)x}{x^T(D_A+D_B)x} .$$

We conclude that

$$\begin{aligned}
\max_x \frac{x^T(|A+B|)x}{x^T(D_A+D_B)x} &= \max_{x\geq 0} \frac{x^T(|A+B|)x}{x^T(D_A+D_B)x} \\
&\leq \max_{y\geq 0} \frac{y^T(|A|+|B|)y}{y^T(D_A+D_B)y} \\
&= \max_y \frac{y^T(|A|+|B|)y}{y^T(D_A+D_B)y} .
\end{aligned}$$

□

This second bound is tighter (or at least as tight) as our first bound (Theorem 2.4.3). This follows from the fact that $\|A\|_2 \leq \||A|\|_2$ for any SPSD A.

2.5. Identifying Factor-Width-2 Matrices

Since FW(2), the set of all matrices with factor width at most two, is a subset of H-matrices, any algorithm to identify H-matrices (generalized diagonally dominant matrices) can easily be adapted to recognize matrices in $FW(2)$. There are many such algorithms, see for instance, [**69, 70**]. Since FW(2) matrices are also SPSD, it may in fact be easier to identify such matrices than general H-matrices.

We show that we can use Theorem 2.4.6 to easily identify matrices with factor-width at most 2. The following theorem shows that FW(2) is exactly the set of symmetric matrices with non-negative diagonals satisfying $\|\mathrm{dn}(|A|)\| \leq 2$.

THEOREM 2.5.1. *Matrix A has factor-width at most 2 if and only if it is symmetric with non-negative diagonals, and satisfies $\|dn(|A|)\| \leq 2$.*

PROOF. ($\Rightarrow$) Let A have factor-width at most 2. Then A is symmetric with non-negative diagonals. By Theorem 2.4.6, $\|\mathrm{dn}(|A|)\| \leq 2$.

($\Leftarrow$) Let A be symmetric with non-negative diagonals satisfying $\|\mathrm{dn}(|A|)\| \leq 2$. Since $\|\mathrm{dn}(|A|)\| = \max_x \frac{x^T|A|x}{x^T D_A x} \leq 2$, it follows that $x^T(2D_A - |A|)x \geq 0$ so $2D_A - |A|$ is positive semidefinite. $2D_A - |A|$ is exactly A's comparison matrix. Since A's comparison matrix is symmetric and positive semidefinite, then it is an M-matrix. Therefore A is an H-matrix. Furthermore, A is symmetric with non-negative diagonals, and therefore A is an H^+-matrix. By Theorem 2.3.6, since A is an H^+-matrix, it has factor-width at most 2. ☐

This result is in fact just a special case of one of many known characterizations of H-matrices:

THEOREM 2.5.2. *The following are equivalent:*
(i) A is a non-singular H-matrix.
(ii) Let $D = diag(A)$. Then $\rho(|I - D^{-1}A|) < 1$.

This theorem was stated in a slightly different (more general) form for M-matrices in [**94**, Thm. 1]. Note that this result holds for all H-matrices (even non-symmetric matrices). Since we allow singular H-matrices, the inequality in case (ii) should be modified to $\rho(|I - D^{-1}A|) \leq 1$. This condition is then equivalent to Theorem 2.5.1 in the SPSD case.

We conclude that our lower bound (Theorem 2.5.1) is always tight for factor width two. We do not know if the bound is tight for factor width three. For large factor widths it is easy to construct examples where the bound is not tight, that is, the factor width is strictly greater than the lower bound.

2.6. Conclusions and Open Problems

We have defined factor width for symmetric matrices and characterized the matrix classes $FW(1)$ and $FW(2)$. An obvious question is, does $FW(k)$ correspond to any known matrix class for other values of k? In particular, what is $FW(3)$? We note that the finite element method naturally produces matrices of low factor width since each element has a small number of degrees of freedom. This indicates that the study of (low) factor width may have practical applications.

Other open problems are the complexity of computing the factor width (Section 2.4), and proving better upper bounds. It could be interesting to study how many columns in V are needed to realize a factor width k decomposition $A = VV^T$. This number can be denoted the "factor width k rank".

Finally, we ask if there is any useful generalization of factor-width for nonsymmetric matrices. A simple but naive choice is to consider factorizations $A = UV^T$ and count the nonzeros in columns of U and V. However, with such a definition any matrix would have "factor width" one since any nonzero a_{ij} in A can be represented by the scaled outer product $a_{ij}e_ie_j^T$.

CHAPTER 3

Combinatorial Characterization of the Null Spaces of Symmetric H-Matrices[1]

3.1. Introduction

This chapter provides a combinatorial characterization of the null spaces[2] of three families of symmetric matrices: Symmetric diagonally-dominant[3] M-matrices (SDDM matrices), symmetric diagonally-dominant matrices (SDD matrices) and symmetric H-matrices with non-negative diagonal entries (which we denote by H^+ matrices). All SDDM matrices are SDD matrices, and all SDD matrices are H^+-matrices, but the converse statements are not true.

The class of H^+ matrices is exactly the class of *factor-width-2* matrices [**15**]. A matrix A is a factor-width-2 matrix if it can be represented as $A = UU^T$, such that each column of U contain at most two non-zeros (U may be rectangular). We characterize the rank and null space of H^+ matrices and of the SDDM and SDD special cases in terms of a width-2 factor U. Given an SDD matrix A, finding a width-2 factor is trivial. For general H^+ matrix, one can use the techniques discussed in Section 2.3 to compute a width-2 factor.

The width-2 factor U of a matrix A can be viewed as an incidence matrix for A's underlying graph. When A is an SDDM matrix, its underlying graph can be viewed as a weighted undirected graph. We show that when A is an SDD matrix, its underlying graph is a signed graph (the term signed graph is taken from [**96**]). We also show that when A is an H^+ matrix, its underlying graph is a gain graph (the term gain graph is also taken from [**96**]; these were previously called voltage graphs [**43, 44**]).

The underlying graph of an H^+ matrix is connected if and only if the matrix is irreducible (does not have a nontrivial block diagonal form).

[1]The results in this chapter were published in "Combinatorial Characterization of the Null Spaces of Symmetric H-Matrices", *Linear Algebra and its Applications 392: 71–90 (2004)* by Doron Chen and Sivan Toledo.

[2]Throughout this chapter, wherever we refer to the characterization of the null space of a matrix, we mean the characterization of a *basis* of the null space of that matrix.

[3]A matrix A is called *diagonally dominant* if $a_{ii} \geq \sum_{j \neq i} |a_{ij}|$. Note that in our definition, the diagonal entries of A are required to be non-negative.

When A is reducible, its null space is the direct sum of the null spaces of its diagonal blocks. We also refer to these null spaces as the null spaces of the connected components of A's graph. Therefore, we analyze the null space of each connected component separately.

Section 2 defines the factor width of a matrix and explains how to find a width-2 factorization of H^+ matrices. Section 3 discusses the connection between factor-width-2 matrices and gain graphs, and shows that the gain graphic matroid is a special case of the vectorial matroid. Section 4 characterizes the rank and null space of irreducible H^+ matrices (and hence, of H^+ matrices in general). Section 5 describes an efficient algorithm to compute the null space of an H^+ matrix given its factor-width-2 representation. This algorithm can be used to efficiently check whether a gain graph is balanced. Section 6 shows how to use our results to accurately determine the rank of SDD matrices and how to solve singular linear systems whose coefficient matrices are SDD. Section 7 contains two simple experimental results. Section 8 contains conclusions and open problems.

3.2. Width-2 Factorization of H^+ Matrices

This chapter analyzes the null spaces of H^+ matrices, given their *factor-width-2* representation.

Factor-width-2 matrices generalize SDD and SDDM matrices: it is a well-known result that an SDD matrix A can be factored into $A = UU^T$ such that the columns of A are scaled edge vectors (either positive or negative) or half-arc vectors, defined as follows:

Definition 3.2.1. The *positive edge vector* $\langle ij\rangle$ has exactly two non-zeros, $\langle ij\rangle_{\min(i,j)} = 1$ and $\langle ij\rangle_{\max(i,j)} = -1$. The *negative edge vector* $\rangle ij\langle$ also has two non-zeros, $\rangle ij\langle_i = 1$ and $\rangle ij\langle_j = 1$. The *vertex vector* $\langle i\rangle$ has exactly one nonzero, $\langle i\rangle_i = 1$. All of these vectors are n-by-1 column vectors, where n is the number of rows in U.

Therefore, SDD matrices are a special case of factor-width-2 matrices: these matrices can be factored into UU^T such that the columns of U contain at most two non-zeros, and in those columns with exactly two non-zeros, the non-zeros are of the same magnitude.

The importance of our study of the class of factor-width-2 matrices stems from the fact that this class is exactly the class of symmetric H-matrices with positive diagonals (H^+ matrices) [**15**]. These matrices occur frequently in engineering and scientific computation [**8, 70**]. Unfortunately, the proof that the class of H^+ matrices is the class of factor-width-2 matrices is not constructive and does not provide an algorithm to obtain factor-width-2 representations of non-diagonally-dominant H^+ matrices.

A useful characteristic of nonsingular H-matrices (see for instance, [**4**, lemma 6.4]) is that they are generalized diagonally dominant, defined as follows:

DEFINITION 3.2.2. A square matrix A is *generalized diagonally dominant* if there is a positive vector $y > 0$ such that for every row i,

$$|a_{ii}|\, y_i \geq \sum_{j \neq i} |a_{ij}|\, y_j \,.$$

The problem of finding a such a vector y is equivalent to the problem of finding a positive diagonal matrix D such that DAD is diagonally-dominant. By solving a linear program, a vector y can be found in time polynomial in the dimension of A, but this is not efficient enough for many applications. The vector y must satisfy the linear constraints $M(A)y \geq 0$, $y \geq 1$, where $M(A)$ is the comparison matrix of A,

$$[M(A)]_{ij} = \begin{cases} +|a_{ij}| & i = j \\ -|a_{ij}| & i \neq j \end{cases} \,.$$

Given such a vector y or such a diagonal matrix D, one can immediately obtain a factor-width-2 representation of A. Several authors proposed iterative algorithms for finding y [**68, 69, 70**], but they prove no useful bounds on the convergence rates or running times of these algorithms.

Transforming A into DAD is equivalent to an operation called *switching* in gain graphs [**79, 96**], in which the gain g of an edge from i to j is replaced by $d_{ii}^{-1} g d_{jj}$. The balance of cycles (discussed later) is invariant under switching.

3.3. Gain Graphs and Factor-Width-2 Matrices

A gain graph is a weighted graph with the property that for each edge connecting vertex i to vertex j, there is also an edge connecting vertex j to vertex i. Because of this property, we consider gain graphs to be *undirected* graphs, although each edge e connecting vertex i and vertex j has two different weights, one in each direction. The weights in the graph are called *gain*, and they have the following property: If the gain of an edge connecting vertex i to vertex j is g, then the gain of the edge connecting vertex j to vertex i is $1/g$. In this chapter we deal with gain graphs over $\Re \setminus \{0\}$. The *gain of a directed path* or cycle is the product of the gains of the edges along the path or cycle. A cycle is *balanced* if its gain is exactly $+1$. In addition to edges with gain, the edge-set of a gain graph may include *half-arcs*, which have only one end point and no gain. As Zaslavsky puts it, half-arcs “trail off into space” [**96**].

Gain graphs (previously called voltage graphs [**43, 44**]) are a generalization of signed graphs [**96**]. A signed graph is a gain graph in which all the gains are ± 1. The notion of balance was introduced by Harary [**55**].

Gain graphs, like undirected unweighted graphs and like signed graphs, induce a matroid. This was stated without a proof in [**96**]. A set of edges is said to be independent if and only if each of its connected components contains at most one cycle (and that one cycle is unbalanced) or one half-arc, but not both.

In this section we prove that this is indeed a matroid by showing that this is a special case of the vector matroid: Given a gain graph matroid, we show how to choose a set of vectors, such that a subset of edges is independent if and only if the corresponding vectors are linearly independent. The vectors that we choose have at most two nonzeros. The matrix whose columns are these vectors can be viewed as a generalized incidence matrix of the gain graph. *The set of these matrices is isomorphic, up to nonzero column scaling, to the set of gain graphs.* This establishes the connection between gain graphs and width-2 factors.

We now define the vectors associated with the edges and half-arcs of gain graphs. Let n be the number of vertices in the gain graph. For each edge e connecting vertex i and vertex j with gain g from i to j (and consequently gain $1/g$ from j to i), we attach an n-by-1 column vector which contains two non-zeros: α in the i-th position and β in the j-th position. We choose α and β so that $\beta/\alpha = -g$. Note the direction we choose for the edge does not affect our choice of vector: had we viewed the same edge as having a gain $1/g$ from j to i, then we would have chosen a vector such that $\alpha/\beta = -1/g$, which is exactly the same property. Also note that given a set of vectors, whether or not they are linearly dependent is not affected by scaling. Therefore, we can choose any vector with the property $-\beta/\alpha = g$ to represent edge e.

DEFINITION 3.3.1. The *generalized edge-vector*$\langle i, j, \alpha \rangle$, $\alpha \neq 0$, corresponding to an edge from i to j with gain α, is the n-vector with the value 1 in position i and the value $-\alpha$ in position j. The *half-arc vector* $\langle i \rangle$ has exactly one nonzero, $\langle i \rangle_i = 1$.

We now prove several simple lemmas about generalized edge vectors and half-arc vectors.

LEMMA 3.3.2. *A generalized edge-vector representing an edge connecting i and j, and a generalized edge-vector representing an edge connecting j and k span a generalized edge-vector representing an edge connecting i and k. The gain of the spanned edge is the product of the two gains.*

PROOF. $\langle i, j, \alpha_1 \rangle + \alpha_1 \langle j, k, \alpha_2 \rangle = \langle i, k, \alpha_1 \alpha_2 \rangle$. □

LEMMA 3.3.3. *The generalized edge vectors corresponding to a directed path of edges from vertex i to vertex j span a generalized edge vector representing an edge connecting i and j. The gain of the spanned edge is the product of the gains of the edges in the path.*

PROOF. This follows from the previous lemma by induction. □

LEMMA 3.3.4. *A generalized edge vector $\langle i, j, \alpha\rangle$ and the half-arc vector $\langle i\rangle$ span vector $\langle j\rangle$.*

PROOF. $-\frac{1}{\alpha}\langle i, j, \alpha\rangle + \frac{1}{\alpha}\langle i\rangle = \langle j\rangle$. □

LEMMA 3.3.5. *The following vectors*

$$\langle i_1, i_2, \alpha_1\rangle, \langle i_2, i_3, \alpha_2\rangle, \ldots, \langle i_{k-1}, i_k, \alpha_{k-1}\rangle, \langle i_k, i_1, \alpha_k\rangle$$

are linearly dependent if and only if $\Pi_{j=1}^{k}\alpha_j = 1$.

PROOF. Consider the following matrix:

$$\begin{pmatrix} 1 & & & & -\alpha_k \\ -\alpha_1 & 1 & & & \\ & -\alpha_2 & \ddots & & \\ & & \ddots & 1 & \\ & & & -\alpha_{k-1} & 1 \end{pmatrix}.$$

Its columns are linearly independent if and only if its determinant is zero. By expanding about the first row, we find that $\det(A) = 1 - \Pi_{j=1}^{k}\alpha_j$. □

The previous lemma is a generalization of a result by Grossman, Kulkarni, and Schochetman [**46**, Theorem 2.1]. They analyzed the case were all the edge vectors have gain -1.

LEMMA 3.3.6. *A cycle is balanced if and only if the vectors corresponding to its edges are linearly dependent.*

PROOF. We assume, without loss of generality (with respect to scaling of edge and half-arc vectors), that the vectors representing the edges are unscaled generalized edge vectors, and apply the proof of the previous lemma. □

The next theorem is the main result of this section, stating the connection between the balanced-cycles matroid of a gain graph, and the associated vectorial matroid.

THEOREM 3.3.7. *Given a gain graph, a subset of its edges is independent if and only if the vectors corresponding to these edges are linearly independent.*

Proof. ($\Longrightarrow$) Suppose that the edges are independent. By definition, this means that each connected component contains at most one cycle (and it is unbalanced) or half-arc, but not both. Suppose to the contrary that the vectors corresponding to the edges are linearly dependent. Then the zero vector can be represented as a linear combination of the vectors. Let $G^* = (V, E^*)$ be the subgraph such that E^* is the set of edges whose coefficients in the linear combination are not zero. We may assume that all the edges of E^* belong to the same connected component, since linear combinations of vectors in different connected component cannot cancel each other out.

We first assume that the G^* does not contain a half-arc. Since G^* is independent, it may contain at most one cycle, which is unbalanced. If i is a leaf, only one generalized edge vector contains a non-zero in position i, so this non-zero cannot be canceled out by the other edges in G^*. Therefore subgraph G^* cannot contain any leaves. Since there are no leaves, E^* must be a cycle. However, since the edges are independent, that cycle must be unbalanced. By lemma 3.3.6, the vectors of the cycle edges are linearly independent, a contradiction.

Now let us assume that G^* does contain a half-arc. The edges of G^* belong to one connected component, so G^* cannot contain a cycle. G^* is a forest, so it must contain at least two leafs. At least one of those leafs is not the end-point of the half-arc. If i is that leaf, only one vector contains a non-zero in position i, so this non-zero cannot be canceled out by the other edges in G^*, a contradiction.

($\Longleftarrow$) Suppose that the vectors are linearly independent. By lemma 3.3.6, a connected component cannot contain a balanced cycle. Suppose a connected component contains an unbalanced cycle. We want to show that no half-arc can exist in that connected component. Let i be a vertex in that cycle. Let k be the length of the cycle. There are k vectors corresponding to the edges of the cycle and they are linearly independent. Therefore vector $\langle i \rangle$ is spanned. Suppose to the contrary that the connected component contains a half-arc, whose endpoint is j. There is a path between vertices i and j, so a generalized edge vector $\langle i, j, \alpha \rangle$ (for some α) can be spanned. Since $\langle i, j, \alpha \rangle$, $\langle i \rangle$ and $\langle j \rangle$ span the zero vector, we have found a non-trivial linear combination representation of the zero vector. Therefore the vectors are linearly dependent, a contradiction. $\square$

3.4. The Null Space of Factor-Width-2 Matrices

This section characterizes the null spaces of factor-width-2 matrices, which are always symmetric and positive semi-definite, but not always diagonally dominant. Grossman, Kulkarni and Schochetman [**45**] analyzed a special case of this result where the factor-width-2 representation contains only unscaled negative edge vectors.

The class of factor-width-2 matrices includes all the matrices that can be factored into $A = UU^T$, such that U has at most two nonzeros per column. The columns of U may have entries that differ in absolute value, so they are not scaled edge vectors. This class clearly does not include all symmetric positive semi-definite matrices. For example, the matrix $(1, 1, 1)^T(1, 1, 1)$ does not belong to this class.

The nonzero structure of U can be viewed as the incidence matrix of the underlying graph G_A of $A = UU^T$, where the columns of U represent edges and half-arcs and the rows represent vertices. The main theorem of this section characterizes the null spaces of factor-width-2 matrices.

THEOREM 3.4.1. *The dimension of the null space is the number of connected components that contain no half-arc vectors and no unbalanced cycles. Furthermore, the null space of A is the direct sum of the* gain vectors *(defined below) of the rank-deficient connected components.*

We symmetrically permute the rows and columns of A into a block diagonal form, where each block $A_1, A_2, \ldots, A_k$ represents a connected component in A's underlying graph. Theorem 3.4.1 follows from the following lemmas.

LEMMA 3.4.2. *Let $A_i = U_iU_i^T$ be an n_i-by-n_i matrix corresponding to a connected component, such that at least one of the columns in U_i is a scaled half-arc vector. Then A_i is full rank.*

PROOF. At least one of U_i's vectors is a scaled half-arc vector. Without loss of generality, let us assume that U_i contains a column which is a scaling of $\langle 1\rangle$. Since there is a path between vertex 1 and each other vertex in the connected component, it follows from Lemma 3.3.3 that for each vertex j in the connected component, U_i spans $\langle 1, j, \alpha\rangle$ for some $\alpha \neq 0$. By Lemma 3.3.4, this vector together with $\langle 1\rangle$ span $\langle j\rangle$. Therefore, the following n_i vectors are spanned: $\langle 1\rangle, \langle 2\rangle, \ldots, \langle n_i\rangle$. These vectors are linearly independent. Therefore A_i is full rank. $\square$

LEMMA 3.4.3. *Let $A_i = U_iU_i^T$ correspond to a connected component with an unbalanced cycle. Then A_i is full rank.*

PROOF. Without loss of generality, let us assume that vertex 1 is in the unbalanced cycle. By lemma 3.3.6, the vectors of that cycle span $\langle 1\rangle$. Since there is a path between vertex 1 and each other vertex in the connected component, by lemma 3.3.3 it follows that for each vertex j in the connected component, a generalized edge-vector representing an edge from vertex 1 to vertex j is spanned. This vector along with $\langle 1\rangle$ span $\langle j\rangle$ (lemma 3.3.4).

Therefore, the following n_i vectors are spanned: $\langle 1\rangle, \langle 2\rangle, \ldots, \langle n_i\rangle$. These vectors are linearly independent. Therefore A_i is full rank. □

LEMMA 3.4.4. *Let $A_i = U_i U_i^T$ correspond to a connected component such that all of the columns of U_i are scaled generalized edge vectors (no scaled half-arc vectors), and such that all the cycles in the corresponding connected component (if any) are balanced cycles. Then the rank of A_i is $n_i - 1$, and its null space is spanned by the gain vector (defined below) of the connected component.*

PROOF. First let us consider a connected component containing no cycles at all. In other words, U_i's columns represent a tree. Matrix U_i is an n_i-by-$(n_i - 1)$ matrix, and so its rank is bounded by $n_i - 1$. Since there is a path between vertex 1 and each of the other vertices j in the connected component we can conclude, as before, that a generalized edge vector, representing an edge from vertex 1 to vertex j, is spanned. These $n_i - 1$ vectors are linearly independent, and so the rank of A_i is exactly $n_i - 1$.

Let y_i be the n_i-by-1 column vector that contains, in the j-th position, the gain of the directed path from vertex j to vertex 1. We call y_i the *gain vector*. For each scaled generalized edge vector $u = \beta\langle k_1, k_2, \alpha\rangle$ in U, the ratio between the values of y_i in positions k_1 and k_2 is $+\alpha$. Therefore, for each $u \in U$, vector u is orthogonal to y_i.

We have found a vector y_i which is orthogonal to each $u \in U_i$. Therefore: $u^T y_i = \overrightarrow{0}$ for each $u \in U_i$. Therefore: $uu^T y_i = \overrightarrow{0}$ for each $u \in U_i$. Hence,

$$A_i y = \sum_{u \in U_i} uu^T y_i = \overrightarrow{0}\ ,$$

i.e. vector y in the null space of A_i.

Now, let us allow cycles in our graph. Given our graph, let us arbitrarily choose any spanning tree of the connected component. That tree has rank $n_i - 1$. Adding edges to the tree can only increase the rank. Let y_i be the gain vector defined above, with regard to the edges in the tree. As we have shown, for each edge vector u corresponding to tree edge, $u^T y_i = 0$.

There are two possibilities:

1. The vectors corresponding to the non-tree edges are spanned by the tree edges, in which case y_i is orthogonal to each vector $u \in U_i$. Therefore $A_i y_i = 0$.

2. The vectors corresponding to the non-tree edges are **not** spanned by the tree edges, in which case A_i is full rank.

Let us consider the edges outside our chosen tree. If one of these edges closes an unbalanced cycle, A_i has full rank. If, on the other hand, all of the cycles in the graph are balanced, then each non-tree edge is spanned by any path of tree edges between its endpoints. In this case y_i is orthogonal to all of the vectors in U_i and is in A_i's null space. $\square$

We have shown that the null space of matrices $U_i U_i^T$ such that U_i's columns contain no half-arc vectors, and U_i's graph is connected and contains no unbalanced cycles, is spanned by the gain vector y_i. The gain vector y_i contains, in the j-th position, the gain of a directed path between vertex j and vertex 1. The vector y_i is well-defined, since for each vertex j the paths from vertex 1 to vertex j all have the same gain (otherwise there would have been an unbalanced cycle in the graph).

The results in this section can be specialized in certain cases. If A is diagonally dominant, its width-2 factorization can be computed in linear time (linear in the number of nonzeros/edges). This factorization contains only edges with gain ± 1 (the corresponding edge vectors were denoted by $\langle ij \rangle$ and $\rangle ij \langle$ in [**13**]) and half-arcs. The gain graph corresponding to A is called a *signed graph* [**96**], the gain of a path/cycle reduces to parity, and the gain vector reduces to a parity vector, containing only ± 1's. Also note that the gain of an edge in a signed graph is the same in both directions.

If A is both diagonally dominant and has only nonpositive off diagonals, all the edge vectors have gain 1. The gain graph corresponds to an unweighted undirected graph. In this case, the gain graphic matroid reduces to a graphic matroid, in which a subset S of edges is independent if and only if it is acyclic (see, for instance, [**25**]). All paths have gain 1, which implies that any cycle in such a graph is balanced. Therefore, the rank of a component is $n_i - 1$ if and only if it has no half arcs, and full rank otherwise. The gain vector in such cases degenerates into a characteristic vector with 1's in the positions corresponding to the component's vertices.

3.5. An Efficient Algorithm for Computing the Null Space of an H^+ Matrix

In this section we describe an efficient algorithm to compute the null space of an H^+ matrix given its width-2 factor. The amount of work that the algorithm performs is linear in the number of edges and vertices.

The algorithm we describe can also be used to efficiently check whether a gain graph is balanced. A gain graph G is called *balanced* if it contains no half arcs and no unbalanced cycles.

By lemmas 3.4.2, 3.4.3 and 3.4.4, a connected component is balanced if and only if it is rank deficient.

The algorithm works as follows. For each connected component, we check whether it contains a half-arc. If so, then the connected component is unbalanced and full-rank. If there is no half-arc, we choose some arbitrary vertex in the component. We call that vertex a *root*. We then construct a spanning tree of the component using a depth-first traversal. In the process, we can easily compute the gain of the path from each vertex to the root. Let us denote the gain of the path between a vertex i and the root r by $\gamma(i)$, and call it the *gain of the vertex*. Then $\gamma(r) = 1$, and the gain of any other vertex i is $\gamma(i) = g(i, \pi_i) \cdot \gamma(\pi_i)$ where π_i denotes i's parent in the tree, and $g(i, \pi_i)$ denotes the gain of the edge (i, π_i). Let y be the vector such that for each vertex i in the connected component $y_i = \gamma(i)$, and for each vertex j not in the connected component, $y_j = 0$. If the connected component is balanced, then y is the *gain vector* of that connected component, and it is in the null space. To determine the rank, we inspect all the edges of the component that are not in the tree. The gain of the path from i to j through the spanning tree is $\gamma(i)/\gamma(j)$. If $(i, j) \in E$ and $g(i, j) \neq \gamma(i)/\gamma(j)$, then (i, j) closes an unbalanced cycle and the component is full rank. If there is no such edge and no half-arcs, the component is rank deficient and $y \in \text{null}(A)$.

If all the connected components are balanced, then the graph is balanced.

3.6. Solving Singular SDD Linear Systems

We have shown how to compute an orthonormal basis for the null space of a H^+ matrix, given its width-2 factor. However, for most H^+ matrices, we do not know how to quickly obtain a width-2 factor (solving a linear feasibility problem can be done in polynomial time, but is unlikely to be fast enough for the applications we consider here). For SDD matrices, however, a width-2 factorization can be computed in time linear in the number of nonzeros in A. Therefore, for SDD matrices we can also compute the rank and a basis for the null space in time linear in the number of nonzeros in A.

In this section we show how to use the characterization and explicit construction of the null space of an SDD matrix to solve linear systems with SDD coefficient matrices. In particular, we show how to address the following issues:

- Determining whether the SDD coefficient matrix A is singular.
- Determining whether a singular SDD linear system $Ax = b$ is consistent, that is, whether b is in the range of A.
- Finding the Cholesky factorization $A = LL^T$, where L is lower triangular, of a singular SDD matrix A.
- Using the Cholesky factorization of a singular SDD matrix to solve a consistent linear system $Ax = b$ or to find the least-squares solution of an inconsistent linear system.

- Using the Conjugate Gradient (CG) algorithm or the Minimum-Residual (MINRES) algorithm to iteratively solve rank-deficient least-squares problems.

The solutions that we propose to these problems are numerically stable and accurate. What we are essentially proposing are combinatorial algorithms that stabilize continuous fixed-precision algebraic computations.

Determining the rank of a general matrix is hard, in the sense that it requires computing at least the small eigenvalues accurately, which is more expensive than solving linear systems. Therefore, linear system solvers usually do not and cannot reliably determine whether a linear system is singular or not.

Our results imply that for general SDD matrices, singularity can be reliably determined in almost linear time. First, we find the connected components that have no negative cycle. This part of the computation is completely sign-based and suffers no rounding errors. Next, we check that each of these components (if any) has a strictly diagonally-dominant row, which corresponds to a half-arc. If one or more components have no negative cycle and no strictly diagonally-dominant rows, the matrix is singular. Determining the diagonal dominance of rows is subject to rounding errors in the summation process. Since all the terms in the summation are positive, however, it is possible to achieve perfect relative accuracy [**58**], [**59**, Chapter 4]. In other words, our algorithm is very accurate and will err only when a diagonal element is larger than the sum of the absolute values of the offdiagonals in a row, but only by a factor of $O(\epsilon_{\text{machine}})$, where $\epsilon_{\text{machine}} \approx 10^{-16}$ in double-precision IEEE floating-point arithmetic. Note that simply requiring that a rank-determination algorithm be backward stable is essentially useless: an algorithm that always returns the dimension of the matrix is backward stable, since every matrix is close to a full-rank matrix. Our algorithm provides accurate answers in the sense that it always reports that singular matrices are singular, and only errs on highly ill conditioned (very close to singular) matrices.

Given a linear system with a singular SDD coefficient matrix, it is often useful (as we'll show later) to determine whether the system is consistent. That is, to determine whether b is in the range of A. We can easily and stably determine this by computing the projection of b on the null space of A. If the norm of the projection is large compared with the norm of b, then b is not in the range of A, otherwise it is (or is close to the range space). Given an orthonormal basis N for the null space of A (note that we always compute an orthogonal basis, so all we need is to normalize the basis), the projection is NN^Tb. Since the projection involves only multiplications by orthonormal matrices, it is backward stable.

The next two problems are addressed by a combination of our techniques and the techniques proposed by Arbenz and Drmač [**3**]. They show how to accurately compute the Cholesky factorization of a semidefinite matrix whose null space is known, and how to use such factorizations to solve consistent linear systems (in fact, their algorithms only require the nonzero structure of the null space). Our contribution is the computation of a basis for the null space of SDD matrices.

We also point out that one can solve square rank-deficient least-squares problems $\min \|Ax - b\|_2$ by projecting b orthogonally onto the range of A and solving the singular but consistent linear system $A\hat{x} = (I - NN^T)b$. The solution $\hat{x}$ of this system of linear equations is the minimum-norm least-squares solution that we seek. Projecting the right-hand-side onto the range of A allows iterative solvers, such as CG [**57**] and MINRES [**73**], to effectively solve the singular consistent system. These methods reliably converge on consistent systems (and then the convergence does not depend on the existence of the zero eigenvalue, only on the nonzero eigenvalues), but they converge very slowly or fail to converge on inconsistent systems. While iterative least-square solvers such as LSQR [**74**] can solve least-squares problems without first projecting the right-hand side, they are often slower than CG and MINRES. Therefore, projecting the right-hand side onto the range of A, which requires a basis for the null space, can be useful. The next section explores these issues experimentally.

3.7. Experimental Results

In the previous section we showed several applications of explicitly constructing the null space of an SDD matrix. In this section we describe numerical experiments that demonstrate two of these applications. The experiments were carried out using MATLAB.

3.7.1. The rank of an SDD matrix. Algorithm 1 constructs an ill-conditioned but full rank SDD matrix A that MATLAB incorrectly classifies as rank deficient. The graph of the matrix is a cycle of size 100, which is exactly but not strictly diagonally dominant. The gain of 99 of the edges is 1 and the gain of the remaining edge is -1. Therefore, the gain of the cycle is -1. This cycle is unbalanced and thus A has full rank no matter how the edges are scaled. The matrix is constructed by computing $A = UU^T$, where U is a scaled incidence matrix. The columns of U corresponding to the positive edges are unscaled positive edge vectors, and the column corresponding to the negative edge is scaled by 1.5×10^5.

The condition number of this matrix, as computed by MATLAB, is approximately 4.56×10^{13}. MATLAB's rank computation is based on computing the number of eigenvalues larger than $n \|A\| \epsilon_{\text{machine}}$, which in this case

Algorithm 1 A full-rank ill-conditioned SDD matrix. There is no similarly-structured *singular* SDD matrix. In other words, A is full rank for any nonzero column scaling of U.

```
n = 100;
U = zeros(n,n);
for i = 1:n-1
   U(i,i)   =  1;
   U(i+1,i) = -1;
end
U(1,n) = 1.5e5;
U(n,n) = 1.5e5;
A = U*U';
rank(A)
```

turns out to be 99. Making the scaling of the negative edge smaller leads to 100 eigenvalues that pass the threshold, so MATLAB reports the rank as 100; making the scaling larger leads to an even smaller reported rank.

This matrix is indeed ill conditioned, so it is very close to some singular matrix. However, by our characterization of the rank of SDD matrices, we know that no SDD matrix with this nonzero structure and with the same element signs is singular. The nonzero structure and the element signs imply that for all numerical element values (as long as diagonal dominance is maintained), the signed graph of the matrix is a negative cycle, so the matrix is nonsingular.

Therefore, if we know that the application in which the matrix arises only produces SDD matrices and that the nonzero and sign structure is accurate (not subject to numerical errors), then we can conclude that the matrix is nonsingular. There is no singular SDD matrix with this nonzero and sign structure, so the singular matrices in the neighborhood of our matrix do not belong to this class. The algorithm that we described in Section 5 determines correctly in linear time that A has full rank. The algorithm computations are completely sign-based and suffer no rounding errors.

3.7.2. Iteratively Solving Square Rank-Deficient Least-Squares Problems. Our second experiment demonstrates the importance of projecting the right hand side of a singular linear system onto the range of the coefficient matrix before using Krylov-subspace linear-equation iterative solvers with short recurrences.

In this experiments we used a 400-by-400 SDD matrix A whose underlying graph is a regular 20-by-20 mesh. The matrix is diagonally dominant but not strictly. The gain of the two edges connected to one of the corner vertices of the mesh is -1, and the gain of all the other edges is 1. The edge vectors are unscaled, so all the offdiagonal elements of A are either 0 or have absolute value 1. Every cycle that contains negative edges must contain the corner vertex and the two negative edges incident to it. Since all the cycles are balanced and there are no half-arcs in this graph, A is rank deficient. The null space of A is spanned by a vector y that contains 1 in the corner vertex and -1 elsewhere.

We selected three random right-hand side vectors b with normal element distributions (using MATLAB's randn routine), and solved the three linear systems of the form $Ax = b$. We used four different Krylov-subspace iterative solvers to solve each linear system, without preconditioning. The iterative solvers were LSQR [**74**], a least-squares solver, and three linear-equation solvers: Conjugate Gradients (CG) [**57**] (see also [**40**]), MINRES [**73**], and GMRES [**80**] with no restarts. We invoked each linear solver on each linear system twice: once with the original random b, and again with an orthogonal projection $(I-yy^T)b$ of b onto the range of A. Thus, we conducted a total of $3 \times 4 \times 2 = 24$ experiments. (We actually conducted more experiments with additional right-hand-sides and additional matrices to ensure that we obtained representative results, but the results of the additional experiments are not shown in this thesis; they were similar to the results shown.)

The results of the experiments are presented in Figures 3.7.1, 3.7.2, and 3.7.3. Each graph plots the norm of the residual as a function of the number of iterations, as computed during the iteration by the iterative algorithm itself. The residuals are not computed directly in every iteration; they are updated, so they may become inaccurate. The graphs also show the completion flag (0 implies convergence, other values imply breakdown) and the 2-norms of the computed solution $\hat{x}$ and of the true residual $b-A\hat{x}$.

The results do not vary qualitatively by the actual right-hand side b. In all cases, all the linear solvers computed an accurate minimum-norm residual-norm minimizer $\hat{x}$ when they were required to solve the consistent linear system $A\hat{x} = (I - yy^T)b$. (We verified that the solution is indeed a minimum-norm minimizer using the pseudo inverse of A.) All the algorithms converged to the desired accuracy, a reduction of the residual norm by a factor of 10^6, within similar numbers of iterations, around 70, except LSQR, which took over 260 iterations to converge.

When the algorithms were required to solve the inconsistent system $A\hat{x} = b$, the behavior of LSQR did not change, but the behavior of the others changed dramatically. The short-recurrence algorithms, CG and MINRES, failed to produce a good residual-norm minimizer. GMRES, on the other hand, returned a residual-norm minimizer $\hat{x}$ when it stopped, but that solution is not a minimum-norm solution (the code returns the approximation that minimizes the residual norm, not necessarily the approximation from the last iteration). None of these three algorithms, including GMRES, could detect convergence even when they achieved it, because no $\hat{x}$ in the Krylov subspaces (a subset of the range of A) can produce a small-enough residual. CG always stopped when the direction vectors overflowed (exit flag 4), while both MINRES and GMRES stopped when they stagnated. The failure to detect convergence caused GMRES to run for much too long.

The fact that GMRES always computed a good solution suggests that CG and MINRES suffer from loss of orthogonality in the direction vectors when the linear system is inconsistent.

Clearly, solving a linear system with a right-hand side that is projected onto the range of A allows the iterative linear solvers to detect convergence more reliably than they otherwise would. Also, it appears that inconsistent linear systems induce a loss of orthogonality and hence failure in short-recurrence algorithms like CG and MINRES.

The iterative least-squares solver, LSQR, produced the same solution whether the system was consistent or not, and within the same number of iterations. However, it performs many more iterations (more than a factor of 3 in this case) than the linear solvers when they solve a consistent system.

There results demonstrate the utility of explicitly computing an orthonormal basis for the null space when solving square least-squares problems. Of the four solvers that we tested, one (LSQR) did not benefit from projection onto the range, but it was slow. Another, GMRES, did compute a residual-norm minimizer, but not a minimum-norm one, and it did not detected convergence. It was also very slow due to full orthogonalization in every step. The best solvers were the short-recurrence solvers, CG and MINRES, when applied to a singular but consistent system; they were fast and computed a minimum-norm residual-norm-minimizer.

3.8. Conclusions and Open Problems

The main results in this chapter are (1) a combinatorial characterization of the null space of H^{+} matrices, which include SDD matrices, (2) efficient algorithms, which rely on this combinatorial characterization, to construct bases for the null spaces of such matrices. We have also demonstrated

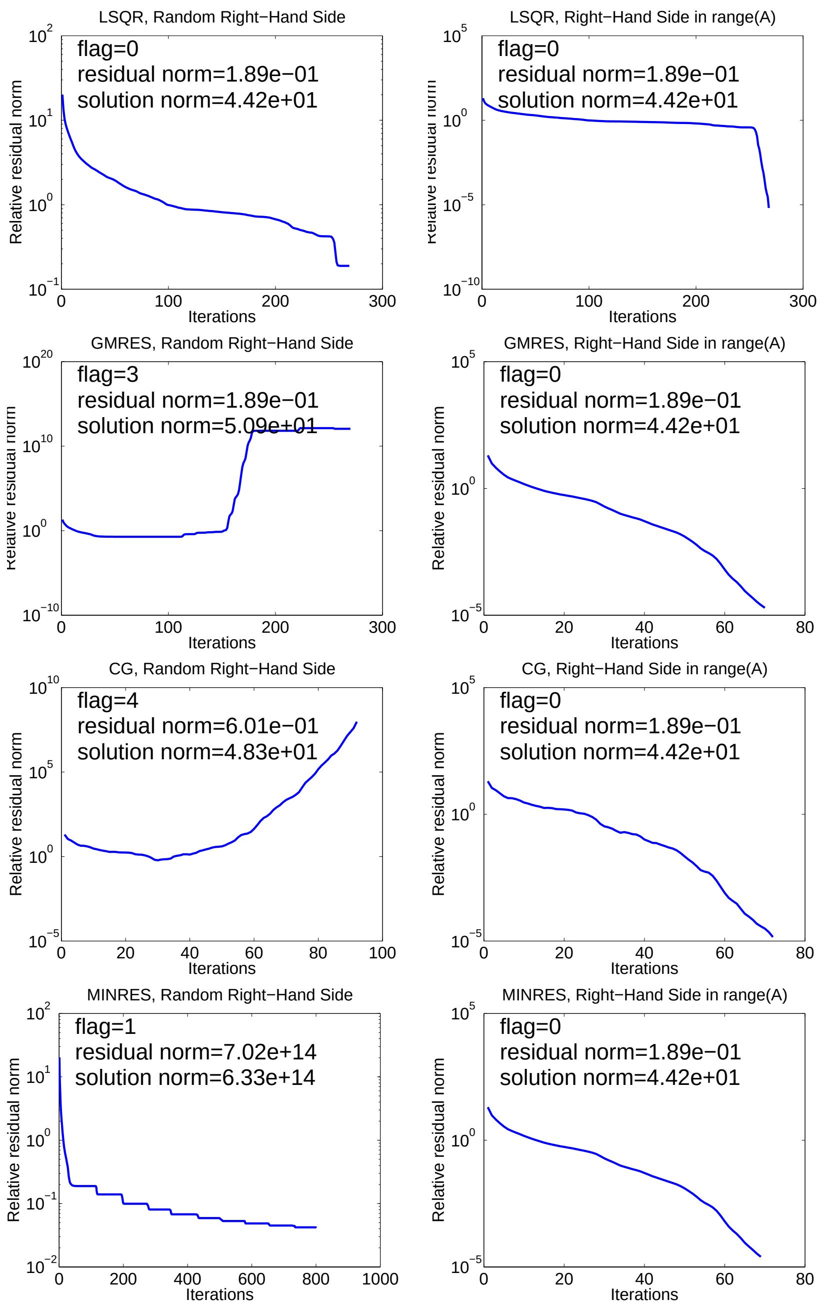


FIGURE 3.7.1. The convergence of Krylov-subspace iterative solvers on a 400-by-400 singular linear system. The coefficient matrix is an SDD matrix representing a 2D mesh. The plots show (top to bottom) the convergence of LSQR, GMRES, CG, and MINRES. In each row, the plot on the left shows the convergence when the right-hand side is random and not in range(A), and the plot on the right shows the convergence when the right-hand-side is projected orthogonally onto range(A).

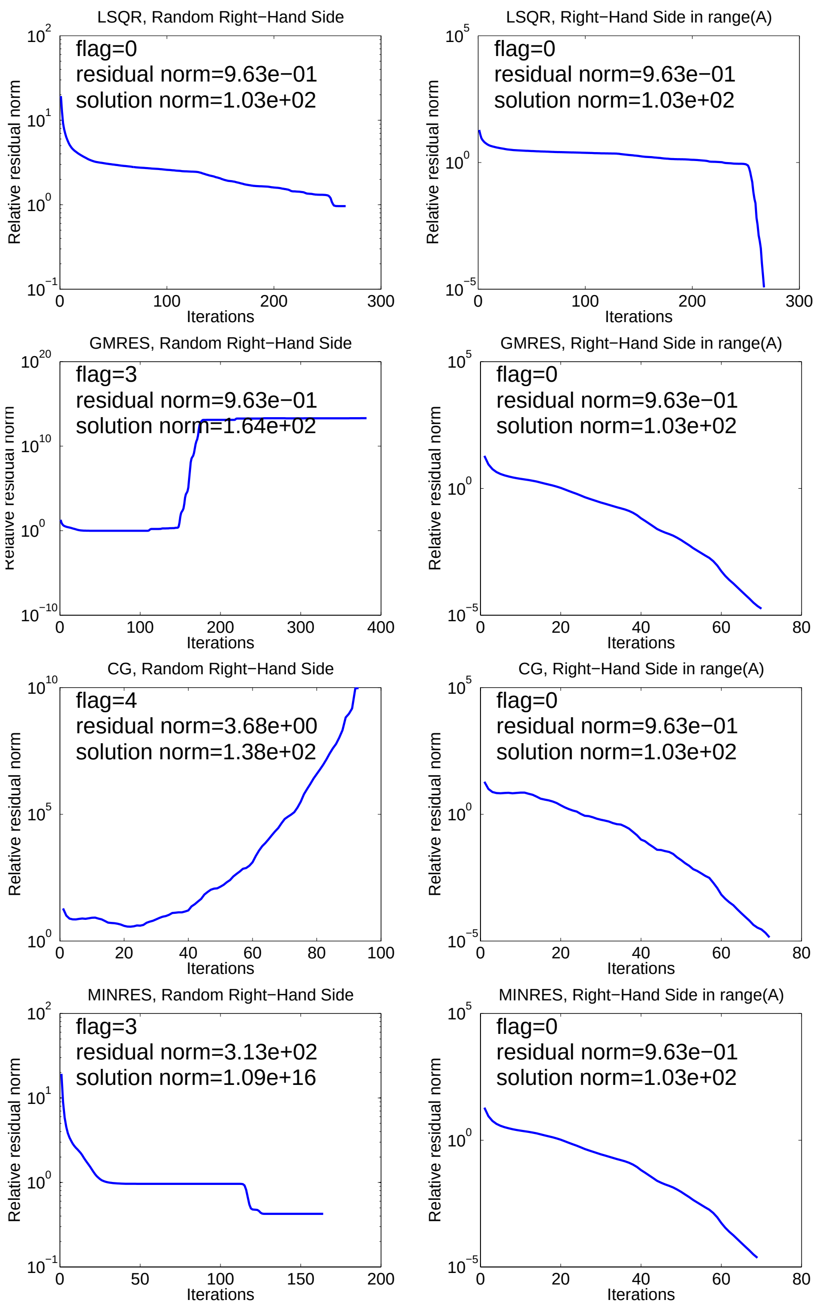


FIGURE 3.7.2. An experiment on an additional right-hand-side.

the utility of these combinatorial algorithms in a number of important numerical-linear-algebra computations.

The chapter extends previous results in spectral graph theory in several directions. First, we study spectral properties of signed and weighted

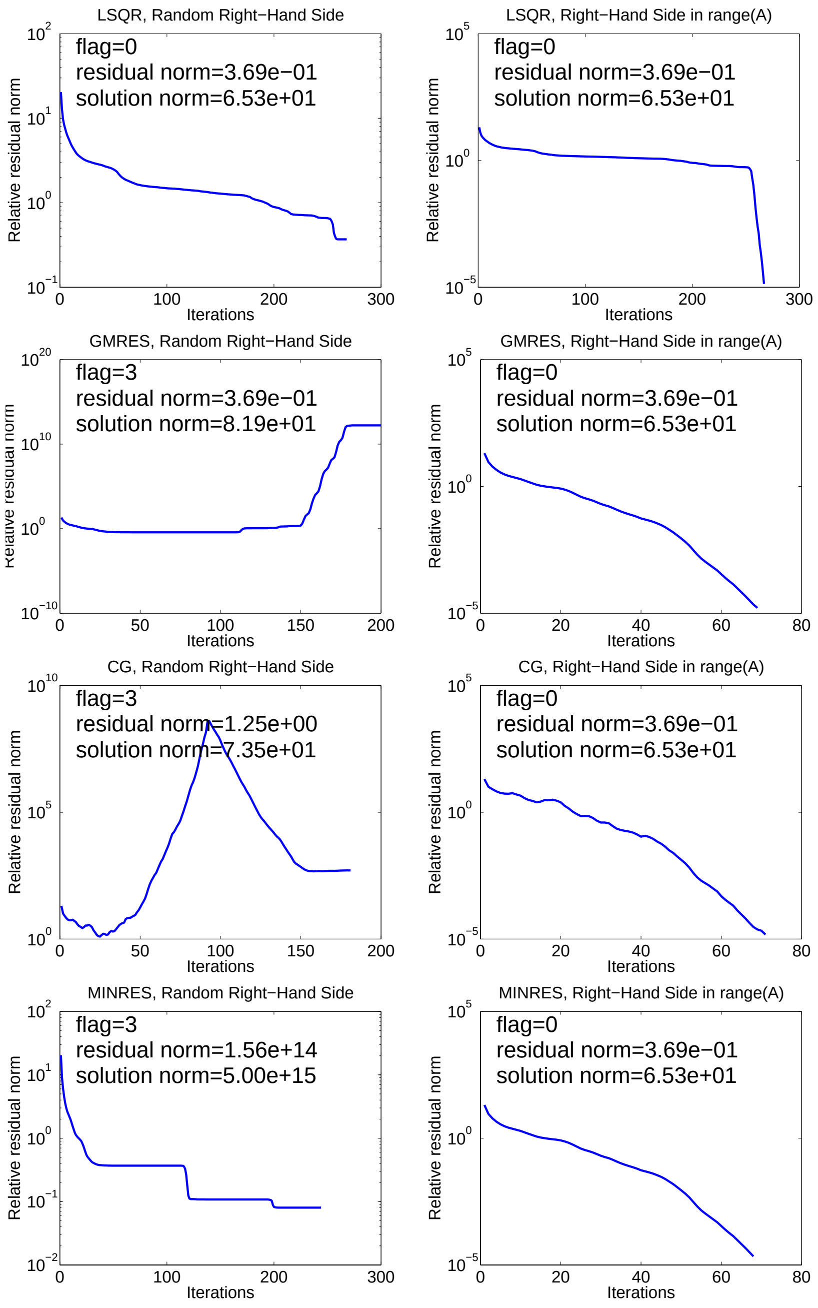


FIGURE 3.7.3. An experiment on an additional right-hand-side.

graphs, namely, the existence of zero eigenvalues. Previously, most of the research in this area has focused on spectral properties of undirected unsigned graphs, using their Laplacian matrices. Second, we study the structure of eigenvectors (those corresponding to zero eigenvalues), rather than estimate or bound eigenvalues. The structure of eigenvectors of Laplacians are important in other applications, such as spectral partitioning of graphs [**48, 87**], but in general Laplacian eigenvectors have not been studied much.

Support preconditioners exploit the simple isomorphism between diagonally-dominant matrices and weighted undirected graphs or signed graphs. This chapter shows that such matrices have very structured null spaces. On the other hand, matrices that arise in applications such as finite-element models of linear elasticity have null spaces which are more complex; the dimension of these null spaces are 3 for two-dimensional problems and 6 for three-dimensional problems. The results of this chapter indicate that such finite-element matrices cannot be approximated by diagonally-dominant matrices; it appears that these matrices cannot be represented as weighted graphs. As a result, it is difficult to apply support theory techniques to this family of finite-element problems.

The chapter raises a number of interesting questions. Perhaps the most important one is whether a given H^+ matrix can be efficiently factored into width-2 factors. Several authors proposed iterative algorithms for this problem [**68, 69, 70**], but there are no useful bounds on their convergence or running time. Finally, more detailed characterizations of the eigenvectors of matrices associated with graphs (such as Laplacians) would be useful in some applications, such as quantization of mesh functions [**85**].

The results of this chapter are related to highly-accurate algorithms for certain classes of ill-conditioned matrices [**28, 29, 67**]. These algorithms solve accurately problems in which the input matrix is ill conditioned; the problem is close to a singular problem. However, the input matrix has a special structure, such as total nonnegativity or Vandermonde. On the manifold of matrices with the same structure, the problem is not ill conditioned. The algorithms solve the problem accurately using a matrix representation that guarantees that all the intermediate matrices in the computation lie on the same structured manifold. Our null-space algorithm for H^+ matrices is similar in that it relies on the factor-width-2 representation of the matrix, a representation that guarantees that the matrix is on the manifold of H^+ matrices.

CHAPTER 4

Obtaining Bounds on the Two Norm of a Matrix from the Splitting Lemma[1]

4.1. Introduction

Support theory [**9, 16**] is a set of tools used to bound the condition numbers of preconditioned systems. Support theory employs two devices to bound the support of a preconditioner: one is the splitting lemma [**9, 42**], and the other is the symmetric product support lemma [**16**]. In this chapter we compare bounds which arise from these two tools, and introduce new bounds.

Conjugate Gradient (CG) is a common iterative algorithm for solving symmetric positive-definite linear systems $Ax = b$. Given a symmetric positive-definite matrix A, the number of iterations needed to reduce the norm of the residual by a constant factor is bounded by the spectral condition number of A. The spectral condition number $\kappa(A)$ is the ratio of the extreme eigenvalues of A, $\kappa(A) = \lambda_{\max}(A)/\lambda_{\min}(A)$. The convergence of CG, as well as of many other iterative solvers, can often be improved by use of a preconditioner B. When using a preconditioner, the number of iterations needed for convergence is bounded by the condition number of the preconditioned system, which is the ratio of the extreme *finite eigenvalues* of the matrix pencil (A, B), defined as follows.

Definition 4.1.1. The number λ is a finite generalized eigenvalue of the matrix pencil (A, B) if there exists a vector x such that $Ax = \lambda Bx$ and $Bx \neq 0$.

Support theory [**9, 16**] is a framework for bounding the condition number of definite and semidefinite preconditioned linear systems. In early support-theory papers [**9, 42**], three main tools were used: the support lemma, the splitting lemma and the congestion-dilation lemma. The support lemma showed how to bound the finite eigenvalues of (A, B) in terms of a number $\sigma(A, B)$ called the *support* of (A, B). The splitting lemma shows that $\sigma(A, B) \leq \max_i \sigma(A_i, B_i)$, where $A = \sum_i A_i$ and $B = \sum_i B_i$.

[1]The results in this chapter were accepted for publication in *Electronic Transactions on Numerical Analysis*, "Obtaining Bounds on the Two Norm of a Matrix from the Splitting Lemma" by Doron Chen, John R. Gilbert and Sivan Toledo.

The congestion-dilation lemma showed how to directly bound $\sigma(A_i, B_i)$ when A_i and B_i are particularly simple: when the graph of A_i consists of a single edge, and the graph of B_i is a simple path between that edge's endpoints[2]. In these early papers all the matrices involved had to be diagonally dominant, but that is irrelevant for our work. In essence, the splitting lemma allowed a complex problem to be broken into simple parts, and the congestion-dilation lemma allowed each part to be analyzed.

Boman and Hendrickson later presented support theory in a completely algebraic framework, which does not refer to graphs, paths, and so on [**16**]. Their framework still used the support lemma, but they replaced much of the rest with a single powerful lemma, the symmetric-product-support lemma. This lemma shows that under suitable conditions on the null spaces of A and B, the finite eigenvalues of the pencil (A, B) are bounded by $\|W\|_2^2$, where $U = VW$, $A = UU^T$, and $B = VV^T$. To apply the lemma, one has to construct a W satisfying these conditions, and to bound its 2-norm. They also show that the bounds that were previously derived by the splitting and congestion-dilation lemmas can be directly obtained by applying their new lemma together with the norm bound $\|W\|_2^2 \leq \|W\|_1 \|W\|_\infty$. It seemed that the splitting lemma was no longer useful.

However, recent results by Spielman and Teng again used the splitting lemma [**88, 89**]. What, then, is the role of the splitting lemma in the Boman-Hendrickson symmetric-product-support framework? This chapter shows that in all its existing applications [**9, 42, 88, 89**], the splitting lemma can be viewed as a mechanism to bound $\|W\|_2^2$ for a given W. We also show that this bound is sometimes tighter than other easily-computed bounds on $\|W\|_2^2$, such as $\|W\|_F^2$ and $\|W\|_1 \|W\|_\infty$.

We also show that certain regular splittings have useful combinatorial interpretations. These interpretations can be exploited to construct and analyze graph algorithms for constructing preconditioners, such as the algorithms in [**13, 42, 88, 89, 93**]. In particular, one of these interpretations was used, with a different proof, in [**88**].

Path embeddings have also been used to bound the smallest nonzero eigenvalue of Laplacian matrices. To do so, one embeds the complete graph in the target graph. Our bounds apply to embeddings of arbitrary graphs, so they are more general. However, special cases of some of our bounds have already been discovered in the more restricted case [**30, 49, 50, 62, 83**].

This chapter is organized as follows. The next section describes the basic results of support theory. Section 4.3 proves the splitting lemma and shows that the symmetric-product-support lemma implies it. Section 4.4

[2]The graph G_A of an n-by-n symmetric matrix A is a weighted graph $G = (V, E, w)$, where $V = \{1, 2, \ldots, n\}$, $E = \{(i, j) \colon A_{ij} \neq 0\}$, and the weight of an edge $w(i, j)$ is $w(i, j) = -A_{ij}$.

describes our main technical tools, orthonormal stretchings and fractional splittings. Section 4.5 proposes two splitting heuristics and shows that they lead to new algebraic and combinatorial bounds on the 2-norm of a matrix. Section 4.6 shows two additional bounds on the 2-norm. Section 4.7 quantifies the behavior of each one of the new norm bounds on an example. In particular, the example shows that the different bounds can be asymptotically different, some tight and some loose. Section 4.8 presents our conclusions.

4.2. Background

This section provides key definitions and known lemmas that we use in the rest of the chapter. We start with the definition of support and with the support lemma.

4.2.1. Support.

Definition 4.2.1. The support $\sigma(A,B)$ of a matrix pencil (A,B) is the smallest number τ such that $\tau B - A$ is positive semidefinite. If there is no such number, we take $\sigma(A,B)=\infty$.

The importance of support numbers stems from the following lemma:

Lemma 4.2.2. *(Support Lemma* [**42**]*) If λ is a finite generalized eigenvalue of (A,B) and B is positive semidefinite and* $\mathrm{null}(A) \subseteq \mathrm{null}(B)$*, then $\lambda \le \sigma(A,B)$. When $\sigma(A,B)$ is finite, the bound is tight.*

Next, we state the key result in Boman and Hendrickson's support framework.

Lemma 4.2.3. *(Symmetric-product-support lemma* [**16**]*) Suppose $U \in \mathbb{R}^{n\times k}$ is in the range of $V \in \mathbb{R}^{n\times p}$. Then*

$$\sigma\left(UU^T, VV^T\right) = \min_W \|W\|_2^2 \ \textit{subject to}\ VW = U \ .$$

4.2.2. Combinatorial Interpretations of Support Bounds.

Lemma 4.2.3 is often used after factoring the n-by-n coefficient matrix A into $A = UU^T$, where U is n-by-m and the preconditioner B into $B = VV^T$, where V is n-by-k (note that there are no special conditions on V and U; they need not be triangular or orthogonal). Typically, the columns of U are *edge vectors* [**13**], i.e. each column of U corresponds to one off-diagonal in the matrix A. Similarly, each column of V corresponds to one off-diagonal of B. This particular factorization is used when A and B are symmetric diagonally-dominant (SDD) matrices. A matrix A that can be decomposed into $A = UU^T$ where U has at most two nonzeros per column is called a *factor-width-2* matrix; the properties of such matrices have been explored in [**15, 22**]. When A is factor-width-2 and has nonpositive offdiagonal entries, it is often called a weighted *Laplacian* matrix.

When A is decomposed into $A = UU^T$ in this way, every column of U corresponds to an edge in the graph G_A of A or to a vertex in G_A, and similarly for $B = VV^T$. In this case, any matrix W satisfying $U = VW$ can be seen as an *embedding* of G_A in G_B. Suppose that column j of U is an edge vector in G_A (otherwise it is a vertex vector). Then $U_{:,j} = VW_{:,j}$ (we use Matlab notation, in which a colon represents all the possible indices). The nonzero entries in $W_{:,j}$ specify a set of edge and vertex vectors in V. We say that the edge in G_A is embedded into this set of edges and vertices in G_B. In some support preconditioners, the embedding is always of edges into simple paths and vertices into single vertices [**9, 42, 93**]. In other support preconditioners, some edges are embedded into up to two cycles and up to two paths [**13**].

In fact, the analysis of the preconditioner usually goes in the other direction. One first shows that given A and B, there exists a "good" embedding of G_A into G_B. Then, from this embedding, one shows how to construct W. Finally, some bound on $\|W\|_2^2$ is proven, and this bounds the finite spectrum of the preconditioned system. Common bounds on $\|W\|_2^2$ that have been used in support preconditioners are $\|W\|_2^2 \le \|W\|_1 \|W\|_\infty$, which has been used implicitly in [**9, 13, 42, 93**], and $\|W\|_2^2 \le \|W\|_F^2$, which is used in [**14**]. In this setup, a good embedding is one that leads to a small norm bound, that is, to a small value for $\|W\|_1 \|W\|_\infty$ or $\|W\|_F^2$.

When W is an embedding of edges into simple paths and of vertices into vertices, the two bounds $\|W\|_2^2 \le \|W\|_1 \|W\|_\infty$ and $\|W\|_2^2 \le \|W\|_F^2$ have useful combinatorial interpretations. The first can be interpreted as product of the worst dilation of a path times the worst congestion through an edge of G_B. Here the dilation of a path π between the endpoints of an edge $e \in G_A$ is defined to be $\sum_{e' \in \pi} |W_{e',e}|$. The congestion through an edge $e' \in G_B$ is defined to be $\sum_{e\colon e' \in \pi(e)} |W_{e',e}|$, where $\pi(e)$ is the path that embeds e. The bound $\|W\|_2^2 \le \|W\|_F^2$ can be interpreted as the sum of all the dilations of all the paths, but with a different definition for dilation, $\sum_{e' \in \pi} W_{e',e}^2$.

4.2.3. The Splitting Lemma.

We now state formally the Splitting Lemma, which is the focus of this chapter.

LEMMA 4.2.4. *(The Splitting Lemma) Let $A = A_1 + A_2 + \ldots + A_q$ and let $B = B_1 + B_2 + \ldots + B_q$. If all A_i and B_i are symmetric positive semidefinite, and if for each i, A_i is in the range of B_i, then $\sigma(A, B) \le \max_i \sigma(A_i, B_i)$.*

Typically, this lemma is used by decomposing A into a sum of rank-1 matrices, each corresponding to one off-diagonal, and by decomposing

B into path matrices, matrices that can be symmetrically permuted to a tridiagonal form, and which have only one nonzero irreducible block.

In the rest of this chapter, we focus on symmetric positive semidefinite matrices, but we do not assume that they are diagonally-dominant (unless specified otherwise).

4.3. The Symmetric Product Support Lemma Implies the Splitting Lemma

What is the relationship of the splitting lemma to the symmetric-product-support lemma? In this section we begin the study of this question. This section shows that the splitting lemma is weaker, in the sense that the symmetric-product-support lemma implies the splitting lemma. The following proof proves Lemma 4.2.4 using a straightforward application of the symmetric-product support lemma.

Proof. Let $A_i = U_i U_i^T$ and $B_i = V_i V_i^T$ be arbitrary symmetric-product decompositions of A_i and B_i. Such a decomposition always exists, given our assumption that both matrices are symmetric positive semidefinite. For example, we can use the scaled eigenvectors of A_i as the columns of U_i, where the scaling is by the square root of the corresponding eigenvalue, and similarly for B_i. Let U_S be the concatenation of $U_1, U_2, \ldots, U_q$ and V_S be the concatenation of $V_1, V_2, \ldots, V_q$. That is,

$$U_S = \begin{pmatrix} U_1 & U_2 & U_3 & \cdots & U_q \end{pmatrix} ,$$

and similarly for V_S. Then $U_S U_S^T = \sum_i U_i U_i^T = \sum_i A_i = A$, and $V_S V_S^T = B$.

By the assumption that A_i is in the range of B_i, the factor U_i must be in the range of V_i. Therefore, there exists a W_i such that $U_i = V_i W_i$.

Let $\hat{W}_i$ be the minimizer of $\min_{W_i} \|W_i\|_2$ subject to $U_i = V_i W_i$. By the Symmetric-Product Support lemma, $\sigma(A_i, B_i) = \left\|\hat{W}_i\right\|_2^2$.

Let

$$W = \begin{pmatrix} \hat{W}_1 & & & \\ & \hat{W}_2 & & \\ & & \ddots & \\ & & & \hat{W}_q \end{pmatrix} .$$

We claim that $V_S W = U_S$.

$$\begin{aligned}
V_S W &= \begin{pmatrix} V_1 & V_2 & V_3 & \cdots & V_q \end{pmatrix} \begin{pmatrix} \hat{W}_1 & & & \\ & \hat{W}_2 & & \\ & & \ddots & \\ & & & \hat{W}_q \end{pmatrix} \\
&= \begin{pmatrix} V_1\hat{W}_1 & V_2\hat{W}_2 & V_3\hat{W}_3 & \cdots & V_q\hat{W}_q \end{pmatrix} \\
&= \begin{pmatrix} U_1 & U_2 & U_3 & \cdots & U_q \end{pmatrix} \\
&= U_S
\end{aligned}$$

The norm of W is equal to $\max_i \left\|\hat{W}_i\right\|$, so by the Symmetric-Product Support Lemma it follows that

$$\sigma(A,B) \le \|W\|_2^2 = \max_i \left\|\hat{W}_i\right\|_2^2 = \max_i \sigma(A_i,B_i) \ .$$

□

4.4. Splitting and Stretching

In this section we show a deeper connection between splitting and the symmetric-product-support lemma. We begin by defining an operation called orthonormal stretching, which allows us to obtain one symmetric-product-support triplet from another. We then show that an important class of splittings, the one which has been used almost exclusively in applications, can be interpreted as an orthonormal stretching. That is, splitting is usually a way to obtain one symmetric-product-support triplet from another, and in particular, to obtain a triplet in which computing $\|W\|_2$ is easy.

4.4.1. Orthonormal Stretching. The *orthonormal stretching* of a symmetric-product-support triplet (U,V,W) is a pair of matrices $(S,\tilde{W})$: a k-by-$\tilde{k}$ matrix S with orthonormal rows ($SS^T = I$), and a matrix $\tilde{W}$ such that $W = S\tilde{W}$.

Why is stretching important for support theory? Because, as the next lemma shows, the triplet $(U, VS, \tilde{W})$ is also a symmetric-product-support triplet, because various norms of $\tilde{W}$ bound the corresponding norms of W and because $\sigma(UU^T,VV^T) \le \left\|\tilde{W}\right\|_2^2$.

Therefore, orthonormal stretching is useful when it allows us to take a symmetric-product-support triplet (U,V,W), for which bounding the norm of W is difficult, and obtain a new triplet $(U,VS,\tilde{W})$, for which bounding the norm of $\tilde{W}$ is easier. The norm of $\tilde{W}$ still bounds $\sigma(UU^T,VV^T)$, which in turn bounds the spectrum of the preconditioned linear system.

LEMMA 4.4.1. *Let $U = VW$, and let S and $\tilde{W}$ be an orthonormal stretching of (U, V, W), so $W = S\tilde{W}$. Then, using the notation $\tilde{V} = VS$, the following hold:*

1. $\tilde{V}\tilde{V}^T = VV^T$
2. $V = \tilde{V}S^T$
3. $U = \tilde{V}\tilde{W}$
4. $\|W\|_2 \leq \left\|\tilde{W}\right\|_2$
5. $\|W\|_F \leq \left\|\tilde{W}\right\|_F$
6. $\|W\|_\infty \leq \sqrt{\tilde{k}}\left\|\tilde{W}\right\|_\infty$*, where $\tilde{k}$ is the number of columns in S.*
7. $\|W\|_1 \leq \sqrt{\tilde{k}}\left\|\tilde{W}\right\|_1$

PROOF. Most of the claims are nearly trivial.

1. $\tilde{V}\tilde{V}^T = (VS)(S^TV^T) = V(SS^T)V = VV^T$
2. $V = VI = V(SS^T) = (VS)S^T = \tilde{V}S^T$
3. $U = VW = V(S\tilde{W}) = (VS)\tilde{W} = \tilde{V}\tilde{W}$
4. $\|W\|_2 = \left\|S\tilde{W}\right\|_2 \leq \|S\|_2 \left\|\tilde{W}\right\|_2 = \left\|\tilde{W}\right\|_2$, because $\|S\|_2 = 1$.
5. To show that $\|W\|_F \leq \left\|\tilde{W}\right\|_F$, we need only compare each column of W to the corresponding column in $\tilde{W}$. Let w_j ($\tilde{w}_j$) be column j of W (of $\tilde{W}$). Then $w_j = S\tilde{w}_j$, so $\|w_j\|_2 = \|S\tilde{w}_j\|_2 \leq \|S\|_2 \|\tilde{w}_j\|_2 = \|\tilde{w}_j\|_2$. Since the norm of each column of $\tilde{W}$ is greater or equal to the norm of the corresponding column of W, $\|W\|_F \leq \left\|\tilde{W}\right\|_F$.
6. Since $W = S\tilde{W}$, we have $\|W\|_\infty = \left\|S\tilde{W}\right\|_\infty \leq \|S\|_\infty \left\|\tilde{W}\right\|_\infty$. We prove the claim by bounding the ∞-norm of S, $\|S\|_\infty = \max_i \sum_{j=1}^{\tilde{k}} |S_{ij}|$. Each row in S is a size-$\tilde{k}$ vector with unit norm. Let sum(v) be the sum of the absolute values of the entries of a vector v. It is easy to show that the maximum of sum(v), over all the vectors v with norm 1, is obtained when all the entries of v are equal. The sum of the entries, for the maximal vector, is the square root of the size of the vector. Therefore, for any row i of S we have $\sum_{j=1}^{\tilde{k}} |S_{ij}| \leq \sqrt{\tilde{k}}$, so $\|S\|_\infty \leq \sqrt{\tilde{k}}$, which proves the claim.
7. Let S' be a completion of S to a $\tilde{k}$-by-$\tilde{k}$ orthonormal matrix. Then $\|S\|_1 \leq \|S'\|_1$ because

$$\|S\|_1 = \max_j \sum_{i=1}^{k} |S_{ij}| = \max_j \sum_{i=1}^{k} \left|S'_{ij}\right| \leq \max_j \sum_{i=1}^{\tilde{k}} \left|S'_{ij}\right| = \|S'\|_1 \ .$$

An equivalent argument to that of claim 6 shows that $\|S'\|_1 \leq \tilde{k}$, which proves the claim.

$\square$

But how do we find a useful orthonormal stretching $(S, \tilde{W})$, a stretching for which the norm of $\tilde{W}$ is easy to bound? The next part of this section shows that in many cases, splitting can be interpreted as such a stretching.

4.4.2. Orthonormal Stretching Via Fractional Splitting. In this section we explain the connection between the orthonormal stretching and splitting.

DEFINITION 4.4.2. A *splitting set* for an n-by-m matrix U and an n-by-k matrix V is a set $D_1, D_2, \ldots, D_m$ of k-row matrices satisfying

- $\sum_{j=1}^{m} D_j D_j^T = I$, and
- for each j, $U_{:,j}$ is in the range of VD_j.

LEMMA 4.4.3. *Let $D_1, D_2, \ldots, D_m$ be a splitting set for U and V. Then $A_j = (U_{:,j})(U_{:,j})^T$ and $B_j = (VD_j)(VD_j)^T$ is a splitting of $A = UU^T$ and $B = VV^T$ in the sense of the splitting lemma. That is, $A = \sum_{i=1}^{m} A_i$, $B = \sum_{i=1}^{m} B_i$, and each A_i is in the range of B_i.*

PROOF. Clearly $A = \sum_{j=1}^{m} A_j$. The sum of the B_j's satisfies

$$\begin{aligned}
\sum_{j=1}^{m} B_i &= \sum_{j=1}^{m} (VD_j)(VD_j)^T \\
&= \sum_{j=1}^{m} VD_jD_j^TV^T \\
&= V\left(\sum_{j=1}^{m} D_jD_j^T\right)V^T \\
&= VV^T \\
&= B\,.
\end{aligned}$$

Since for each j, $U_{:,j}$ is in the range of VD_j, each A_j is in the range B_j. $\square$

A splitting set can be difficult to construct, due to the second condition in its definition. But a W satisfying $U = VW$ offers an opportunity to create a special family of splitting sets.

DEFINITION 4.4.4. Let U be an n-by-m matrix, V an n-by-k matrix, and W a k-by-m matrix such that $U = VW$. A *fractional splitting set* for U, V, and W is a set of k-by-k diagonal matrices $D_1, D_2, \ldots, D_m$ satisfying the following conditions.

- The indices of the nonzero diagonal entries in D_j is the set $\{i \colon W_{i,j} \neq 0\}$.
- $\sum_{j=1}^{m} D_j D_j^T = I$.

LEMMA 4.4.5. *A fractional splitting set is a splitting set.*

PROOF. We need to show that for each j, $U_{:,j}$ is in the range of VD_j. Let D_j^+ be the Moore-Penrose pseudo-inverse of D_j. Since D_j is diagonal, D_j^+ is also diagonal, with

$$\left(D_j^+\right)_{ii} = \begin{cases} (D_j)_{ii}^{-1} & (D_j)_{ii} \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

(see, for instance, [**40**, Section 5.5.4]). The matrix $D_j D_j^+$ is diagonal with zeros and ones on the diagonal, with ones in positions that correspond to nonzeros in $W_{:,j}$. Therefore, $D_j D_j^+ W_{:,j} = W_{:,j}$, so $VD_j D_j^+ W_{:,j} = VW_{:,j}$. Therefore,

$$\begin{aligned} VD_j\left(D_j^+ W_{:,j}\right) &= VW_{:,j} \\ &= U_{:,j} \ , \end{aligned}$$

which proves the claim. □

We now show that a fractional splitting set defines not only a splitting of A and B, but also an orthonormal stretching of V and W. We begin by showing how to derive S from the D_j's.

LEMMA 4.4.6. *Let $D_1, D_2, \ldots, D_m$ be a splitting set. Then the concatenation S of the D_j's, $S = \begin{pmatrix} D_1 & D_2 & D_3 & \cdots & D_m \end{pmatrix}$ has orthonormal rows (the concatenation matrix S consists of the columns of D_1 followed by the columns of D_2, and so on).*

PROOF. $SS^T = \sum_{j=1}^{m} D_j D_j^T = I$. □

Clearly, the proof of the previous lemma only relies on one of the two conditions that splitting sets must satisfy.

Next, we show how to construct $\tilde{W}$. The example in the beginning of Section 4.5 illustrates this construction.

LEMMA 4.4.7. *Let $D_1, D_2, \ldots, D_m$ be a fractional splitting set for some U, V, and W, and let S be defined as in Lemma 4.4.6. Let*

$$\tilde{W}_{:,j} = \begin{pmatrix} 0\ ; & 0\ ; & \cdots & 0\ ; & D_j^+ W_{:,j}\ ; & 0\ ; & \cdots & 0\ ; & 0 \end{pmatrix} ,$$

where 0 *denotes the k-by-1 zero vector. (We use the Matlab notation: a semicolon denotes stacking blocks, so $(A\ ;\ B) = (A^T\ B^T)^T$.) That is, in the first column of $\tilde{W}$ the first k elements are $D_1^+ W_{:,1}$ and the rest are zeros. The second column of $\tilde{W}$ starts with k zeros, then the elements of $D_2^+ W_{:,2}$, followed by zeros, and so on. Then $W = S\tilde{W}$.*

Proof. We prove the lemma column by column,

$$\begin{aligned}
\left(S\tilde{W}\right)_{:,j} &= S\tilde{W}_{:,j} \\
&= \begin{pmatrix} D_1 & D_2 & \cdots & D_{j-1} & D_j & D_{j+1} & \cdots & D_{m-1} & D_m \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ D_j^{+}W_{:,j} \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \\
&= \left(\sum_{i\neq j} D_i \cdot 0\right) + D_j D_j^{+} W_{:,j} \\
&= 0 + W_{:,j} \; .
\end{aligned}$$

□

An important benefit of using a fractional splitting to define an orthonormal stretching $(S, \tilde{W})$ is that the 2-norm of $\tilde{W}$ is easy to compute.

Lemma 4.4.8. *Let $(S, \tilde{W})$ be an orthonormal stretching defined by a fractional splitting as in Lemmas 4.4.6 and 4.4.7. Then*

$$\left\|\tilde{W}\right\|_2 = \max_{j=1}^{m} \left\|\tilde{W}_{:,j}\right\|_2 \; .$$

Proof. By the construction of $\tilde{W}$ given in Lemma 4.4.7 it is clear that its columns are orthogonal. □

In general, splittings and symmetric products are not isomorphic. A splitting $A = \sum A_i$ and $B = \sum B_i$ does not define symmetric products $A = UU^T$ and $B = VV^T$, not even implicitly. Also, a symmetric-product-support triplet does not define a splitting. But we have shown that an important class of splittings does define an orthonormal stretching, a way to get one symmetric-product-support triplet from another.

In most of the applications of the splitting lemma [**9, 42, 88, 89**], there is also a symmetric-product representation of A and B, a representation using edge and vertex vectors, and an implicit W. Furthermore, in these applications, the splitting of A and B can almost always be interpreted as a fractional splitting set $D_1, \ldots, D_m$ of the symmetric-product factors U and V. In all of these cases, the splitting can be interpreted as an orthonormal stretching of a symmetric-product-support triplet.

Before we conclude this section, we show that for orthonormal stretchings derived from fractional splitting sets, one of the norm-bounds on $\tilde{W}$ can be tightened.

LEMMA 4.4.9. *Let $(S, \tilde{W})$ be an orthonormal stretching of (U, V, W), derived from a fractional splitting set, as defined in Lemmas 4.4.6 and 4.4.7. Then* $\|W\|_1 \le \left\|\tilde{W}\right\|_1$.

PROOF. We show that $\|S\|_1 \le 1$. By definition, $\|S\|_1 = \max_j \sum_{i=1}^k |S_{ij}|$. By the construction of S in Lemma 4.4.6, each column of S is a column of one of the D_j's. Each column of D_j has exactly one nonzero. Because $\sum_j D_j D_j^T = I$, that nonzero must be no larger than 1 in absolute value. Therefore, each column of S has exactly one nonzero no larger than 1 in absolute value, which proves the claim that $\|S\|_1 \le 1$. Therefore $\|W\|_1 = \left\|S\tilde{W}\right\|_1 \le \|S\|_1 \left\|\tilde{W}\right\|_1 \le \left\|\tilde{W}\right\|_1$. $\square$

Note that whenever each column of S has a single nonzero, $\|W\|_1 \le \left\|\tilde{W}\right\|_1$, even if $\tilde{W}$ was not derived from a fractional splitting set. In particular, the 1-norm bound given in the previous lemma may hold even when the columns of $\tilde{W}$ are not orthogonal. When $(S, \tilde{W})$ are obtained from a fractional splitting set, the bound $\|W\|_2^2 \le \|W\|_1 \|W\|_\infty \le \sqrt{k} \left\|\tilde{W}\right\|_1 \left\|\tilde{W}\right\|_\infty$ is not particularly useful, because we can directly compute $\left\|\tilde{W}\right\|_2$. But in more general cases this bound may be useful.

4.5. How to Split

The choice of D_j's in a fractional splitting can have a profound influence on how close $\left\|\tilde{W}\right\|_2$ is to $\|W\|_2$. We use a fractional splitting because $\left\|\tilde{W}\right\|_2$ is easy to compute and it bounds $\|W\|_2$. In this section we show that a poor choice of D_j's can lead to $\left\|\tilde{W}\right\|_2$ being so large that it teaches us nothing about $\|W\|_2$. We also suggest two simple and efficient heuristics to find splittings with a small $\left\|\tilde{W}\right\|$. From one of these heuristics we obtain two combinatorial bounds on support preconditioners; one of these bounds was already suggested in a more general form by Spielman and Teng **[88]** using an entirely different proof, and the other is new.

4.5.1. An Example. We first show that if the choice of splitting is poor, then the resulting norm bound is useless. Let

$$A = \begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix}, \quad U = \begin{pmatrix} 1 & 0 & 1 \\ -1 & 1 & 0 \\ 0 & -1 & -1 \end{pmatrix}$$
$$B = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}, \quad V = \begin{pmatrix} 1 & 0 \\ -1 & 1 \\ 0 & -1 \end{pmatrix}.$$

To complete U and V to a symmetric-product-support triplet, we use

$$W = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

The 2-norm of W is $\|W\|_2 = \sqrt{3}$.

We now split A and B using the following fractional support set,

$$D_1 = \begin{pmatrix} \epsilon & 0 \\ 0 & 0 \end{pmatrix}, \quad D_2 = \begin{pmatrix} 0 & 0 \\ 0 & 1/\sqrt{2} \end{pmatrix}, \quad D_3 = \begin{pmatrix} \sqrt{1-\epsilon^2} & 0 \\ 0 & 1/\sqrt{2} \end{pmatrix},$$

where $\epsilon > 0$ is small. Therefore,

$$S = \begin{pmatrix} D_1 & D_2 & D_3 \end{pmatrix} = \begin{pmatrix} \epsilon & 0 & 0 & 0 & \sqrt{1-\epsilon^2} & 0 \\ 0 & 0 & 0 & 1/\sqrt{2} & 0 & 1/\sqrt{2} \end{pmatrix}.$$

We now construct $\tilde{W}$ according to lemma 4.4.7, starting with the pseudo-inverses,

$$D_1^+ = \begin{pmatrix} 1/\epsilon & 0 \\ 0 & 0 \end{pmatrix}, \quad D_2^+ = \begin{pmatrix} 0 & 0 \\ 0 & \sqrt{2} \end{pmatrix}, \quad D_3^+ = \begin{pmatrix} 1/\sqrt{1-\epsilon^2} & 0 \\ 0 & \sqrt{2} \end{pmatrix},$$

so

$$\tilde{W} = \begin{pmatrix} 1/\epsilon & & \\ 0 & & \\ & 0 & \\ & \sqrt{2} & \\ & & 1/\sqrt{1-\epsilon^2} \\ & & \sqrt{2} \end{pmatrix}.$$

For small ϵ, $\left\|\tilde{W}\right\|_2 = 1/\epsilon$ is arbitrarily large, so it is not a useful bound on $\|W\|_2 = \sqrt{3}$.

Clearly, $\epsilon = 1/\sqrt{2}$ is a better choice than a small ϵ, yielding a $\left\|\tilde{W}\right\|_2 = 2$, still not completely tight, but better. In this case, a fractional splitting can actually achieve $\left\|\tilde{W}\right\|_2 = \|W\|_2 = \sqrt{3}$. Let

$$S = \begin{pmatrix} \sqrt{1/3} & 0 & 0 & 0 & \sqrt{2/3} & 0 \\ 0 & 0 & 0 & \sqrt{1/3} & 0 & \sqrt{2/3} \end{pmatrix},$$

so

$$\tilde{W} = \begin{pmatrix} \sqrt{3} & & \\ 0 & & \\ & 0 & \\ & \sqrt{3} & \\ & & \sqrt{3/2} \\ & & \sqrt{3/2} \end{pmatrix} = \begin{pmatrix} \sqrt{3} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \sqrt{3} & 0 \\ 0 & 0 & \sqrt{3/2} \\ 0 & 0 & \sqrt{3/2} \end{pmatrix}.$$

(We show $\tilde{W}$ twice, with and without all the zeros, to emphasize the structure of its columns.) Still, the ϵ-example shows that a poor splitting yields useless bounds.

4.5.2. The Rowwise Heuristic. When $W = S\tilde{W}$, row i in W is a linear combination of a set of rows in $\tilde{W}$, where the coefficients of the linear combinations come from row i in S. When $(S, \tilde{W})$ is an orthonormal stretching derived from a fractional splitting set, the row sets that combine to form rows of W are disjoint. This is a consequence of the fact that columns in S have no more than a single nonzero. Therefore such stretchings map disjoint sets of rows of $\tilde{W}$ to the rows of W.

The first heuristic that we propose finds a fractional splitting that ensures that the 2-norm of each nonzero row of $\tilde{W}$ that maps onto row i in W is exactly $\|W_{i,:}\|_2$. This ensures, in a heuristic way, that $\tilde{W}$ is not too large.

Here is another way to interpret this heuristic. Under an orthonormal stretching derived from a fractional splitting set, each nonzero in W is mapped into a nonzero in $\tilde{W}$. The rowwise heuristic ensures that all the nonzeros in $\tilde{W}$ that map to nonzeros in row i of W have the same magnitude.

LEMMA 4.5.1. *Let us define m diagonal matrices, such that the (i,i) value in the j-th matrix is*

$$(D_j)_{i,i} = \frac{W_{i,j}}{\|W_{i,:}\|_2}.$$

Then the D_j's are a fractional splitting set for W.

PROOF. Clearly, the indices of the nonzero diagonal entries in D_j is the set $\{i\colon W_{i,j} \neq 0\}$.

We need to show that $\sum_{j=1}^{m} D_j D_j^T = I$. A sum of diagonal matrices is also diagonal. Therefore, $\sum_{j=1}^{m} D_j D_j^T = \sum_{j=1}^{m} D_j^2$ is diagonal. We need only show that each diagonal entry in $\sum_{j=1}^{m} D_j^2$ is 1. By definition, the (i,i) entry in D_j is $W_{i,j}/\left\|W_{i,:}\right\|_2$. The (i,i) entry in $\sum_{j=1}^{m} D_j^2$ is

$$\sum_{j=1}^{m} \frac{W_{i,j}^2}{\|W_{i,:}\|_2^2} = \frac{1}{\|W_{i,:}\|_2^2} \cdot \sum_{j=1}^{m} W_{i,j}^2 = \frac{1}{\|W_{i,:}\|_2^2} \cdot \|W_{i,:}\|_2^2 = 1 \ .$$

□

We now prove that this splitting preserves the 2-norm of rows in W. We need the following notation: $\eta(i,j) = (j-1)\cdot k + i$ (for $1 \le i \le k$ and $1 \le j \le m$). Matrix S is a concatenation of the D_j matrices. Therefore, S is a concatenation of m k-by-k matrices. $\eta(i,j)$ is the index of the column in S corresponding to the i-th column in D_j.

Lemma 4.5.2. *Let $\tilde{W}$ be an orthonormal stretching of W derived using the rowwise fractional-splitting heuristic. If $W_{i,j} \neq 0$, then $\left\|\tilde{W}_{\eta(i,j),:}\right\|_2 = \left|\left(D_j^+ W_{:,j}\right)_i\right| = \|W_{i,:}\|_2$. Therefore, the norm of each nonzero row in $\tilde{W}$ is the norm of some row in W, and for each row in W, there is at least one row with the same norm in $\tilde{W}$.*

Proof. Rows in $\tilde{W}$ have at most a single nonzero, when $\tilde{W}$ is constructed from a fractional splitting. Therefore, for each nonzero row, the norm of a row is the absolute value of the single nonzero element in that row (the norm of a zero row is zero). All nonzeros in column j of $\tilde{W}$ are entries of $D_j^+ W_{:,j}$. Therefore, for each $1 \le i \le k$ and $1 \le j \le m$, $\left\|\tilde{W}_{\eta(i,j),:}\right\|_2 = \left|\left(D_j^+ W_{:,j}\right)_i\right|$. If $W_{i,j} \neq 0$, then $\left(D_j\right)_{i,i} = W_{i,j}/\left\|W_{i,:}\right\|_2 \neq 0$. Therefore $\left(D_j^+\right)_{i,i} = \left\|W_{i,:}\right\|_2 / W_{i,j}$ and thus:

$$\begin{aligned}
\left(D_j^+ W_{:,j}\right)_i &= \left(D_j^+\right)_{i,i} \cdot W_{i,j} \\
&= \frac{\|W_{i,:}\|_2}{W_{i,j}} \cdot W_{i,j} \\
&= \|W_{i,:}\|_2 \ .
\end{aligned}$$

This proves the lemma, because

$$\left\|\tilde{W}_{\eta(i,j),:}\right\|_2 = \left|(D_j^+ W_{:,j})_i\right| = \|W_{i,:}\|_2 \ .$$

□

We now prove a new bound on $\|W\|_2$, a bound which we later show has a useful combinatorial interpretation.

LEMMA 4.5.3. *Let W be a k-by-m real matrix. Then*

$$\|W\|_2^2 \le \max_j \sum_{i\colon W_{i,j}\neq 0} \|W_{i,:}\|_2^2 = \max_j \sum_{i\colon W_{i,j}\neq 0} \sum_{c=1}^{m} W_{i,c}^2 \,.$$

PROOF. We stretch W orthonormally using the rowwise fractional-splitting heuristic. Then $\|W\|_2^2 \le \left\|\tilde{W}\right\|_2^2$, and the columns of $\tilde{W}$ are orthogonal, so

$$\left\|\tilde{W}\right\|_2^2 = \max_j \left\|\tilde{W}_{:,j}\right\|_2^2 \,.$$

All that remains to show is that $\left\|\tilde{W}_{:,j}\right\|_2^2 = \sum_{i\colon W_{i,j}\neq 0} \sum_{c=1}^{m} W_{i,c}^2$.

$$\begin{aligned}\left\|\tilde{W}_{:,j}\right\|_2^2 &= \left\|D_j^{+} W_{:,j}\right\|_2^2\\ &= \sum_{i=1}^{k} \left(D_j^{+} W_{:,j}\right)_i^2\\ &= \sum_{i\colon W_{i,j}\neq 0} \|W_{i,:}\|_2^2 \,.\end{aligned}$$

The last equality is by lemma 4.5.2. □

LEMMA 4.5.4. *Let W be a k-by-m real matrix. Then*

$$\|W\|_2^2 \le \max_i \sum_{j\colon W_{i,j}\neq 0} \|W_{:,j}\|_2^2 = \max_i \sum_{j\colon W_{i,j}\neq 0} \sum_{r=1}^{k} W_{r,j}^2 \,.$$

PROOF. The previous lemma, applied to W^T, proves the claim, since $\left\|W^T\right\|_2 = \|W\|_2$. □

4.5.3. A Combinatorial Interpretation and the Spielman-Teng Bound. Given a symmetric-product-support triplet (U, V, W), a column of W can be viewed as an embedding of a column of U into the columns of V, since $U_{:,j} = VW_{:,j}$. The nonzero elements in the column of W specify a *generalized path* in V that supports the column in U. When U and V have at most two nonzeros per column, (that is, UU^T and VV^T have factor-width 2), they can be viewed as the weighted incidence matrices of G_{UU^T} and G_{VV^T}. In that case, a generalized path defined by a column of W is, indeed, an edge set, although this edge set may not form a simple path. We now define the dilation of a path that supports a column of U, and the congestion caused by the paths that utilize a column of V.

DEFINITION 4.5.5. Let (U, V, W) be a symmetric-product-support triplet. We say that column i in V *supports* column j in U if $W_{i,j} \neq 0$. The *2-dilation* of a column j in U is

$$\mathrm{dil}_2(U, V, W, j) = \mathrm{dil}_2(j) = \|W_{:,j}\|_2^2 \ .$$

The *2-congestion* of a column i in V is

$$\mathrm{cong}_2(U, V, W, i) = \mathrm{cong}_2(i) = \|W_{i,:}\|_2^2 \ .$$

These definitions, together with Lemmas 4.5.3 and 4.5.4, give the following results.

LEMMA 4.5.6. *Let (U, V, W) be a symmetric-product-support triplet. Then*

$$\|W\|_2^2 \leq \max_j \sum_{i \text{ supports } j} cong_2(i) \ .$$

The next lemma is a special case of the Spielman-Teng Support Theorem [**88**, Theorem 2.1]. Their proof technique, however, is different. The result stated in Lemma 4.5.6 is, to the best of our knowledge, new.

LEMMA 4.5.7. *Let (U, V, W) be a symmetric-product-support triplet. Then*

$$\|W\|_2^2 \leq \max_i \sum_{i \text{ supports } j} dil_2(j) \ .$$

4.5.4. The Frobenius Heuristic. Another approach to fractionally splitting W is to minimize the Frobenius norm of $\tilde{W}$. The Frobenius heuristic defines m diagonal matrices, such that the (i, i) value in the j-th matrix is

$$(D_j)_{i,i} = \frac{\sqrt{|W_{i,j}|}}{\sqrt{\sum_{c=1}^{m} |W_{i,c}|}} \ .$$

LEMMA 4.5.8. *The preceding definition of the D_j's defines a fractional splitting set.*

PROOF. As in Lemma 4.5.1, we need to prove that $\sum_{j=1}^{m} D_j D_j^T = I$. Matrix $\sum_{j=1}^{m} D_j D_j^T = \sum_{j=1}^{m} D_j^2$, being the sum of diagonal matrices, is also diagonal. We need only show that each diagonal entry in $\sum_{j=1}^{m} D_j^2$ is 1. By definition, the (i, i) entry in D_j is $\sqrt{|W_{i,j}|}/\sqrt{\sum_{c=1}^{m} |W_{i,c}|}$. The (i, i) entry in $\sum_{j=1}^{m} D_j^2$ is

$$\sum_{j=1}^{m} \frac{|W_{i,j}|}{\sum_{c=1}^{m} |W_{i,c}|} = \frac{1}{\sum_{c=1}^{m} |W_{i,c}|} \cdot \sum_{j=1}^{m} |W_{i,j}| = 1 \ .$$

□

LEMMA 4.5.9. *The Frobenius heuristic minimizes* $\left\|\tilde{W}\right\|_F$ *over all fractional splittings of* W.

PROOF. We prove the lemma in two steps. We first show that the minimization problem can be broken up into k separate problems, each involving one row of S and one row of W. We then show that the Frobenius heuristic minimizes the contribution of each row to $\left\|\tilde{W}\right\|_F$, and is, hence, an optimal Frobenius-norm minimization strategy.

The nonzero elements of $\tilde{W}$ are the nonzero elements of the vectors $D_j^+ W_{:,j}$ for $j = 1, \ldots, m$. The i-th element of the vector $D_j^+ W_{:,j}$ is $\left(D_j^+\right)_{i,i} W_{i,j}$. If $W_{i,j} = 0$ then $\left(D_j^+\right)_{i,i} W_{i,j} = 0$, otherwise $\left(D_j^+\right)_{i,i} W_{i,j} = W_{i,j} / \left(D_j\right)_{i,i}$.

Therefore, the Frobenius norm of $\tilde{W}$ is

$$\left\|\tilde{W}\right\|_F^2 = \sum_{j=1}^{m} \sum_{i: W_{i,j} \neq 0} \left(\frac{W_{i,j}}{\left(D_j\right)_{i,i}} \right)^2 .$$

In this double sum, the outer summation is over columns of $\tilde{W}$, and inner summation is over the nonzeros in a particular column. Each nonzero of W appears exactly once in the summation. We can change the order of summation so that we sum over rows of W,

$$\left\|\tilde{W}\right\|_F^2 = \sum_{i=1}^{k} \sum_{\substack{j=1 \\ W_{i,j} \neq 0}}^{m} \left(\frac{W_{i,j}}{\left(D_j\right)_{i,i}} \right)^2 . \tag{4.5.1}$$

To minimize the Frobenius norm, we minimize this sum subject to the constraints

$$\sum_{\substack{j=1 \\ W_{i,j} \neq 0}}^{m} \left(D_j\right)_{i,i}^2 = 1 \text{ for all } i = 1, \ldots, k .$$

Since we have a separate constraint for each one of the inner sums in Equation 4.5.1 (for each row of W), the global minimum of the Frobenius norm is achieved when each one of the inner sums is minimized.

We now turn to the second part of the proof, showing that the heuristic does minimize each inner sum. The inner sum minimization is equivalent to finding the vector $(x_1, \ldots, x_m)$ that minimizes $\sum_{i=1}^m (c_i/x_i)^2$ subject to $\sum_{i=1}^m x_i^2 = 1$. The vector c corresponds to the nonzero elements in the ith row of W and the vector x to the corresponding elements of S. We prove by induction on m that the minimum is $\left(\sum c_i\right)^2$ and that it is achieved at

$$x_i = \frac{\sqrt{|c_i|}}{\sqrt{\sum_{j=1}^{m} |c_j|}} .$$

The inductive claim is actually slightly stronger. We prove that when the constraint is replaced by $\sum_{i=1}^{m} x_i^2 = z$ for some $z > 0$, the minimum is $z^{-1}\left(\sum c_i\right)^2$ and that it is achieved at

$$x_i = \frac{\sqrt{z\,|c_i|}}{\sqrt{\sum_{j=1}^{m} |c_j|}} .$$

For $m = 1$ the only choice for x_1 is $x_1 = \sqrt{z}$ and it is easy to verify that the claim holds. Assume that the claim holds for $m-1$. For any value of $0 < x_m < \sqrt{z}$, the minimum of the sum $\sum_{i=1}^{m-1}(c_i/x_i)^2$ subject to $\sum_{i=1}^{m-1} x_i^2 = z - x_m^2$ is, by the inductive claim, $(z - x_m^2)^{-1}\left(\sum_{i=1}^{m-1} c_i\right)^2$. The total minimization problem, then, is to minimize

$$f\left(x_m\right) = \frac{\left(\sum_{i=1}^{m-1} c_i\right)^2}{\left(z - x_m^2\right)} + \frac{c_m^2}{x_m^2}$$

subject to $0 < x_m < \sqrt{z}$. The derivative of this objective function with respect to x_m^2 is

$$\frac{\partial f}{\partial\left(x_m^2\right)} = \frac{\left(\sum_{i=1}^{m-1} c_i\right)^2}{\left(z - x_m^2\right)^2} - \frac{c_m^2}{\left(x_m^2\right)^2} .$$

It is easy to verify that the derivative vanishes at

$$x_m = \frac{\sqrt{z\,|c_m|}}{\sqrt{\sum_{j=1}^{m} |c_j|}} .$$

Clearly, this value of x_m satisfies $0 < x_m < \sqrt{z}$, so it solves the constrained minimization problem. Given this value of x_m, we have

$$\left(z - x_m^2\right) = z\left(\frac{\sum_{j=1}^{m-1} |c_j|}{\sum_{j=1}^{m} |c_j|}\right) .$$

By induction, for $i < m$, the optimal value of x_i under the constraint $\sum_{i=1}^{m-1} x_i^2 = z - x_m^2$ is achieved at

$$x_i = \frac{\sqrt{\left(z - x_m^2\right)|c_i|}}{\sqrt{\sum_{j=1}^{m-1} |c_j|}} = \frac{\sqrt{z\,|c_i|}}{\sqrt{\sum_{j=1}^{m} |c_j|}} .$$

This concludes the inductive claim and the entire proof. $\square$

An alternative way to prove the second part of the proof, proposed by Dan Spielman, uses Lagrange multipliers.

PROOF. The proof that the minimization problem can be broken into k independent subproblems is the same as in the first proof. We now show that $x_i = \sqrt{|c_i| / \sum_j |c_j|}$ is a minimizer of $\sum_{j=1}^{m} (c_j/x_j)^2$ subject to $\sum_{j=1}^{m} x_j^2 = 1$.

Let $f(x_1, \ldots, x_m, \lambda) = \sum_{j=1}^{m} \left(\frac{c_j}{x_j}\right)^2 + \lambda \left(\sum_{j=1}^{m} x_j^2 - 1\right)$. The minimizer satisfies

$$\begin{array}{rcccl} 0 & = & \frac{\partial f}{\partial x_i} & = & -2 \cdot \frac{c_i^2}{x_i^3} + 2\lambda x_i \ , \\ 0 & = & \frac{\partial f}{\partial \lambda} & = & \sum_{j=1}^{m} x_j^2 - 1 \ . \end{array}$$

It follows that $c_i^2 = \lambda x_i^4$, therefore $x_i^2 = |c_i| / \sqrt{\lambda}$. Since $\sum_{j=1}^{m} x_j^2 = 1$ it follows that

$$\begin{array}{rcl} \sum_{j=1}^{m} \frac{|c_j|}{\sqrt{\lambda}} & = & 1 \\ \frac{1}{\sqrt{\lambda}} \sum_{j=1}^{m} |c_j| & = & 1 \\ \sqrt{\lambda} & = & \sum_{j=1}^{m} |c_j| \ . \end{array}$$

Since $x_i^2 = |c_i| / \sqrt{\lambda}$ it follows that

$$x_i^2 = \frac{|c_i|}{\sum_{j=1}^{m} |c_j|} \ ,$$

and thus

$$x_i = \frac{\sqrt{|c_i|}}{\sqrt{\sum_{j=1}^{m} |c_j|}} \ .$$

□

Like the rowwise heuristic, the Frobenius heuristic also produces new algebraic bounds on $\|W\|_2$. These bounds and their proofs were discovered by Dan Spielman [**86**]. Before we state and prove the bounds, we prove an auxiliary result.

LEMMA 4.5.10. [**86**] *Let $\tilde{W}$ be an orthonormal stretching of W derived using the Frobenius fractional-splitting heuristic. If $W_{i,j} \neq 0$, then* $\left\| \tilde{W}_{\eta(i,j),:} \right\|_2 = \left| \left(D_j^+ W_{:,j} \right)_i \right| = \sqrt{|W_{i,j}|} \cdot \sqrt{\sum_{c=1}^{m} |W_{i,c}|}$.

PROOF. As in lemma 4.5.2, for each $1 \leq i \leq k$ and $1 \leq j \leq m$, $\left\| \tilde{W}_{\eta(i,j),:} \right\|_2 = \left| \left(D_j^+ W_{:,j} \right)_i \right|$. If $W_{i,j} \neq 0$, then $(D_j)_{i,i} = \sqrt{|W_{i,j}|} / \sqrt{\sum_{c=1}^{m} |W_{i,c}|} \neq 0$. Therefore $\left(D_j^+ \right)_{i,i} = \sqrt{\sum_{c=1}^{m} |W_{i,c}|} / \sqrt{|W_{i,j}|}$

and thus:

$$\begin{aligned}\left(D_j^+ W_{:,j}\right)_i &= \left(D_j^+\right)_{i,i} \cdot W_{i,j}\\ &= \frac{\sqrt{\sum_{c=1}^m |W_{i,c}|}}{\sqrt{|W_{i,j}|}} \cdot W_{i,j}\\ &= \sqrt{|W_{i,j}|} \cdot \sqrt{\sum_{c=1}^m |W_{i,c}|}\,.\end{aligned}$$

This proves the lemma, because each row of $\tilde{W}$ has at most a single nonzero, so

$$\left\|\tilde{W}_{\eta(i,j),:}\right\|_2 = \left|(D_j^+ W_{:,j})_i\right| = \sqrt{|W_{i,j}|} \cdot \sqrt{\sum_{c=1}^m |W_{i,c}|}\,.$$

□

We now state and prove new bounds on $\|W\|_2$.

LEMMA 4.5.11. [**86**] *Let W be a k-by-m matrix. Then*

$$\|W\|_2^2 \le \max_j \sum_{i:W_{i,j}\neq 0} |W_{i,j}| \cdot \left(\sum_{c=1}^m |W_{i,c}|\right)\,.$$

PROOF. We stretch W orthonormally using the Frobenius fractional-splitting heuristic. Then $\|W\|_2^2 \le \left\|\tilde{W}\right\|_2^2$, and the columns of $\tilde{W}$ are orthogonal, so

$$\left\|\tilde{W}\right\|_2^2 = \max_j \left\|\tilde{W}_{:,j}\right\|_2^2\,.$$

All that remains to show is that $\left\|\tilde{W}_{:,j}\right\|_2^2 = \sum_{i:W_{i,j}\neq 0} |W_{i,j}|\cdot(\sum_{c=1}^m |W_{i,c}|)$. We have

$$\begin{aligned}\left\|\tilde{W}_{:,j}\right\|_2^2 &= \left\|D_j^+ W_{:,j}\right\|_2^2\\ &= \sum_{i=1}^k \left(D_j^+ W_{:,j}\right)_i^2\\ &= \sum_{i=1}^k |W_{i,j}| \cdot \left(\sum_{c=1}^m |W_{i,c}|\right)\\ &= \sum_{i:W_{i,j}\neq 0} |W_{i,j}| \cdot \left(\sum_{c=1}^m |W_{i,c}|\right)\,.\end{aligned}$$

The equality of the second and third lines is by lemma 4.5.10. □

LEMMA 4.5.12. [**86**] *Let W be a k-by-m matrix. Then*

$$\|W\|_2^2 \le \max_i \sum_{j: W_{i,j} \neq 0} |W_{i,j}| \cdot \left(\sum_{r=1}^{k} |W_{r,j}| \right) .$$

PROOF. The previous lemma, applied to W^T, proves the claim, since $\left\|W^T\right\|_2 = \|W\|_2$. □

The bounds in lemmas 4.5.3 and 4.5.11 are structurally similar. Both bound $\|W\|_2^2$ using an expression of the form

$$\|W\|_2^2 \le \max_j \sum_{i: W_{i,j} \neq 0} \sum_{c=1}^{m} g\left(W_{i,j}, W_{i,c}\right) .$$

In lemmas 4.5.3 we have $g\left(W_{i,j}, W_{i,c}\right) = W_{i,c}^2$ and in lemma 4.5.11 we have $g\left(W_{i,j}, W_{i,c}\right) = |W_{i,j}| \cdot |W_{i,c}|$. In both cases the maximum is over sums of functions of the same nonzero elements of W. A similar relationship exists between lemmas 4.5.4 and 4.5.12.

We note that there exist matrices W for which applying the Frobenius heuristic gives a smaller $\left\|\tilde{W}\right\|_2$ than the rowwise heuristic, and that there are matrices for which the rowwise heuristic gives a smaller $\left\|\tilde{W}\right\|_2$. In general, neither of the two is an optimal 2-norm minimization strategy.

4.6. Gram Bounds on the Two Norm

In this section we suggest two additional bounds on the 2-norm of W. In one particular case, these two bounds are equivalent to the bounds proved in Lemmas 4.5.6 and 4.5.7.

LEMMA 4.6.1. *For any matrix W,*

$$\|W\|_2^2 \le \left\|WW^T\right\|_1 = \left\|WW^T\right\|_\infty .$$

PROOF. For all matrices W, $\|W\|_2^2 = \left\|WW^T\right\|_2$. For any matrix A, we have $\|A\|_2^2 \le \|A\|_1 \|A\|_\infty$. In particular, $\left\|WW^T\right\|_2^2 \le \left\|WW^T\right\|_1 \left\|WW^T\right\|_\infty$. Since WW^T is symmetric, $\left\|WW^T\right\|_1 = \left\|WW^T\right\|_\infty$. This concludes the proof. □

Similarly,

LEMMA 4.6.2. *For any matrix W,*

$$\|W\|_2^2 \le \left\|W^TW\right\|_1 = \left\|W^TW\right\|_\infty .$$

Consider the case where A and B are symmetric and diagonally-dominant matrices, and the weights of all the edges in A's and B's underlying graphs are 1. Given an embedding of the edges of A into simple paths in B, all the entries of W are either 0, 1 or -1. In this case, the dilation of an edge is exactly the length of its supporting path. It is easy to see that, in this case, the $\left\|WW^T\right\|_1$ bound is the same as the bound given in lemma 4.5.7. Similarly, the $\left\|W^TW\right\|_1$ bound is the same as the bound in lemma 4.5.6.

In more complex cases, however, the two norm bounds given in this section are not equivalent to the bounds in Lemmas 4.5.6 and 4.5.7.

4.7. An Example

The example that we present in this section shows that the new norm bound given in Lemma 4.5.6 is sometimes asymptotically tighter than all the other norm bounds that we are aware of. In this example, Lemma 4.5.6 tightly bounds the two norm, while all the other bounds are asymptotically loose. In particular, the $\|W\|_F$-norm bound, the $\|W\|_1\|W\|_\infty$-norm bound, the $\left\|WW^T\right\|_1$-norm bound and its equivalents, and the bound in Lemma 4.5.4 are all loose.

Consider the 1-by-2 block matrix $W = (W'|I)$, where I is the $(2m+3)$-by-$(2m+3)$ identity for some m, and where W' is the $(2m+3)$-by-$(m+1)$ matrix

$$W' = \left(\begin{array}{cccc}
\sqrt{m} & 1 & \cdots & 1 \\
\sqrt{m} & & & \\
 & \sqrt{m} & & \\
 & & \ddots & \\
 & & & \sqrt{m} \\
\sqrt{m} & & & \\
 & \sqrt{m} & & \\
 & & \ddots & \\
 & & & \sqrt{m}
\end{array}\right) .$$

The matrix W corresponds to an embedding of the edges of the graph shown in Figure 4.7.1 onto paths in the same graph, but without the dashed edges. Because the graph without the dashed edges is a tree, each edge in the original graph is supported by exactly one simple path.

We can prove the following norm bounds on W. We omit the proofs.

- $\|W\|_2^2 = 4m + 1$.
- $\|W\|_F^2 = 2m^2 + 6m + 3 = \Theta(m^2)$.
- $\|W\|_1\|W\|_\infty = (\sqrt{m} + m + 1)(3\sqrt{m}) = \Theta(m^{1.5})$.
- $\left\|W^TW\right\|_1 = 3m + (m + 3)\sqrt{m} = \Theta(m^{1.5})$.

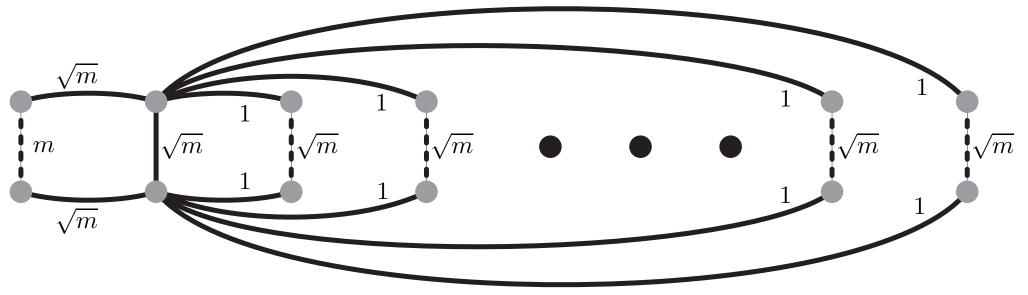


FIGURE 4.7.1. A weighted graph G_{UU^T} with $2m+4$ vertices ($m+2$ on the top and $m+2$ on the bottom). The dashed edges are in G_{UU^T} but not in G_{VV^T}. The edge weights given are the nonzero coefficients of the corresponding columns of U and V; for example, the edge with weight m corresponds to a column $(m, -m, 0, \ldots, 0)^T$ in U.

- $\left\|WW^T\right\|_1 = 4m + 2m\sqrt{m} + 1 = \Theta(m^{1.5})$.
- $\max_j \sum_{i:\, W_{i,j}\neq 0} \|W_{i,:}\|_2^2 = 4m + 3 = \Theta(m)$.
- $\max_i \sum_{j:\, W_{i,j}\neq 0} \|W_{:,j}\|_2^2 = \|W'\|_F^2 + 1 = 2m^2 + 4m + 1 = \Theta(m^2)$.
- $\max_j \sum_{i:\, W_{i,j}\neq 0} |W_{i,j}| \cdot \left(\sum_{c=1}^{m} |W_{i,c}|\right) = \sqrt{m}(\sqrt{m} + m + 1) + 2\sqrt{m}(\sqrt{m} + 1) = \Theta(m^{1.5})$.
- $\max_i \sum_{j:W_{i,j}\neq 0} |W_{i,j}| \cdot \left(\sum_{r=1}^{k} |W_{r,j}|\right) = \sqrt{m}\cdot 3\sqrt{m} + m(1+2\sqrt{m}) + 1 = \Theta(m^{1.5})$.

For large m, none of these bounds on the 2-norm are tight, except for one, $4m+3$, which is not only asymptotically tight, but is off by only a small additive constant.

4.8. Conclusions

We have shown that applying the splitting lemma to the analysis of support-graph preconditioners can be viewed as a mechanism to bound the norm of a matrix W. The mechanism works by orthonormally stretching W into a larger matrix $\tilde{W}$ whose 2-norm bounds that of W but is easier to compute.

In doing so, we have unified the "old-style" support theory, in which the analysis of a preconditioner usually starts by splitting, and the "new-style" support theory, which relies on the symmetric-product-support lemma, usually without splitting.

We also presented six new bounds on the 2-norm of the matrix, given in Lemmas 4.5.3, 4.5.4, 4.5.11, 4.5.12, 4.6.1, and 4.6.2. One of the four was already given by Spielman and Teng, but not in the form of a norm bound. Four of the new bounds have useful combinatorial interpretations. Special cases of some of our new bounds were previously used to bound the smallest nonzero eigenvalue of Laplacian matrices [**30, 49, 50, 62, 83**].

Viewing splitting as a way of bounding $\|W\|_2$ using $\left\|\tilde{W}\right\|_2$ leads to systematic splitting strategies that aim to minimize some other norm of $\tilde{W}$. We propose two such strategies in this chapter; one is a heuristic which preserves in $\tilde{W}$ the 2-norm of rows of W, and another which minimizes the Frobenius norm of $\tilde{W}$. Both are analytically and computationally simple.

We have also shown that one of the new bounds can be asymptotically tighter than all the other norm bounds that we are aware of. The problem of ranking the bounds by tightness, or showing that they cannot be ranked, remains open.

CHAPTER 5

Algebraic Analysis of High-Pass Quantization[1]

5.1. Introduction

High-pass mesh quantization is a compression technique for three-dimensional polygonal meshes. This technique assumes that the connectivity of the mesh has already been encoded, and that it is, therefore, known to both the encoder and to the decoder. The goal of the technique is to compactly encode the coordinates of the vertices of the mesh. High-pass quantization, which was recently proposed by Sorkine, Cohen-Or, and Toledo [**85**], encodes the coordinates by applying a linear transformation based on the mesh Laplacian to the coordinates and quantizing the transformed coordinates. The decoder then applies another transformation to recover an approximation of the original coordinates from the quantized transformed data. The advantage of encoding the transformed coordinates lies in the fact that they can be aggressively quantized without introducing visually disturbing errors. As shown in [**85**], the quantization error is mostly comprised of low-frequency bands, while the high-frequency components of the reconstructed surface are preserved. Since humans are usually more sensitive to changes in lighting (or normals) and the local high-frequency details of the surface, low-frequency errors are perceived as less visible.

Applying the mesh Laplacian to the coordinate prior to quantization is a bad idea. In high-pass quantization, we do not apply the Laplacian itself, but rather a carefully constructed operator derived from it. The construction aims to control two aspects of the compression and decompression process. First, the Laplacian is singular, and it tends to be ill conditioned on large meshes. The singularity reflects the fact that Laplacian-transformed coordinates do not prescribe the absolute positioning of the mesh in space; this singularity is easy to handle. But the ill conditioning is more difficult to handle. If not addressed, the ill conditioning leads to a decompression

[1]The results in this chapter were accepted for publication in *ACM Transactions on Graphics*, "Algebraic Analysis of High-Pass Quantization" by Doron Chen, Daniel Cohen-Or, Olga Sorkine, and Sivan Toledo. The results in section 5.6 are primarily due to Sorkine and Cohen-Or. The results in sections 5.4 and 5.5 are primarily due to Chen and Toledo.

operator with a large norm, which greatly amplifies even small quantization errors. Our construction addresses the ill conditioning using so-called anchor points in the mesh. Anchor points are mesh points whose original coordinates are included in the encoded (transformed) mesh coordinates. In this chapter we show how to estimate the condition of the Laplacian-derived operator from the connectivity of the mesh and the identity of the anchors. By adding anchors appropriately, we control the norm of the quantization error.

The shape of the error is the other aspect of the compression process that our construction aims to control. Laplacian coordinates, with or without anchors, can be thought of as smoothness constraints that the decompressor tries to satisfy. A small Laplacian coordinate at a mesh vertex implies that the mesh is smooth around that vertex, and a large Laplacian coordinate implies local roughness. Anchors add constraints on absolute positioning of the anchor vertices to the decompression process. The key to high-pass quantization is to use both smoothness and absolute positioning constraints at the anchors. This is what controls the shape of the quantization error.

Sorkine et al. [**85**] presented the algebraic framework of high-pass quantization, together with a partial argument that explained why it works well. More specifically, that argument showed how the eigenvalues of the linear transformations that the encoder and the decoder apply affect the quantization error. However, the analysis in [**85**] is incomplete: (1) the analysis there only applies to one class of matrices (so-called k-anchor invertible Laplacians) but not to the matrices that are actually used in the algorithm (k-anchor rectangular Laplacians); (2) the eigenvalues of the transformations are not analyzed, and (3) the effect of rounding errors on encoding and decoding is not analyzed. In this chapter we rectify the deficiencies of [**85**]. In particular, we extend the analysis to show that the singular values of rectangular Laplacians can bound the encoding error, we present bounds on the eigenvalues and singular values of Laplacians, and we bound the effect of rounding errors on the method. The bounds that we derive for the singular/eigen values and for the encoding and rounding errors are given in terms of topological properties of the mesh, so these bounds are relatively easy to estimate. These topologically-derived bounds are also useful for selecting anchors, the extra vertices whose coordinates are used to decode the mesh.

We complement this analysis with a new anchor-selection algorithm and with experimental results. The new algorithm selects anchor points so as to minimize our theoretical error bound. The new experimental results further strengthen the claims in [**85**] concerning the effectiveness of high-pass quantization, and they show how our theoretical bounds relate to the actual encoding errors. It should be noted that the bounds on the condition number of k-anchor rectangular Laplacians are useful for evaluating any methods based on such matrices, such as mesh editing with differential coordinates [**71**].

5.2. Background: Mesh compression

Mesh compression involves two problems that are usually solved, at least conceptually, separately: the mesh *connectivity* encoding and the *geometry* encoding. While state-of-the-art connectivity encoding techniques are extremely effective [**1, 51, 65, 91**], compressing the geometry remains a challenge. The encoded geometry is, on average, at least five times larger than the encoded connectivity, even when the coordinates are pre-quantized to 10–12 bits. Finer quantization for higher precision increases the importance of effective geometry encoding even further.

Earlier works on geometry compression employed prediction-correction coding of quantized vertex coordinates. Linear predictors are usually used; the most common one is known as the parallelogram predictor [**91**]. The displacements are compressed by some entropy encoder. Chou and Meng [**23**] use vector quantization instead to gain speed.

Recent compression methods represent the mesh geometry using effective bases, such as the spectral basis [**63**] which generalizes the Fourier basis functions to irregular connectivity, or the wavelet basis [**66**]. The spectral encoding of Karni and Gotsman [**63**] preserves the original connectivity of the mesh, and relies on the fact that it is known both to the encoder and the geometry decoder (this is also the case with the high-pass quantization method). The mesh compression framework of Khodakovsky et al. [**66**] requires semi-regular remeshing of the input mesh. While their method achieves excellent compression ratios, it is not connectivity-lossless, which thus puts this work in a somewhat different category. In many cases it is desirable to preserve the original connectivity of the mesh, especially when it carefully models certain features and is particularly adapted to the surface geometry. For a recent survey on mesh compression techniques the reader is referred to [**2**].

5.3. Background: High-pass quantization

This section reviews the high-pass mesh quantization method [**85**] that this chapter analyzes.

Sorkine et al. [**85**] proposed a new approach to geometry quantization that works for meshes with arbitrary connectivity. Instead of directly quantizing the Cartesian coordinates, which may lead to errors that damage the high-frequency details of the surface, they proposed to first transform the coordinates to another space by applying the Laplacian operator associated with the mesh topology. The transformed coordinates are called "δ-coordinates". The quantization is applied to the δ-coordinates, and the geometry of the mesh can be restored on the decoder side by solving a linear least-squares system defined by the extended Laplacian matrix, which is described later in this section. They showed that introducing high-frequency errors by quantizing the δ-coordinates results in *low-frequency* errors in the reconstructed Cartesian coordinates, and argued that low-frequency displacements in the surface geometry are less noticeable to the human eye than high-frequency displacements.

5.3.1. Quantization Errors under Linear Transformations. Quantizing a vector x with continuous coefficients introduces an error q_x, where $x+q_x$ is the quantized vector. In this section we show how to control the spectral behavior of the error using linear transformations. We assume that a simple fixed-point quantization is used, so that the maximum quantization error $\max_i |q_i|$ is bounded by the expression $2^{-p}(\max_i x_i - \min_j x_j)$, using p-bit quantized coefficients.

Suppose that instead of quantizing the input vector x, we first transform x into a vector Ax using a nonsingular matrix A, and then quantize Ax. We denote the quantization error by q_{Ax}, so that the new quantized vector is $Ax + q_{Ax}$. The elements of the quantized vector are now discrete, as are those of $x+q_x$. We can recover an approximation of x from this representation, by multiplying the quantized vector by A^{-1}:

$$A^{-1}(Ax + q_{Ax}) = x + A^{-1}q_{Ax}.$$

The error in this approximation is $A^{-1}q_{Ax}$, and we will shortly see that under certain conditions, it behaves quite differently than q_x.

Assume that A has an orthonormal eigen-decomposition $AU = U\Lambda$, where U is unitary and Λ is diagonal. This assumption is satisfied when A is real and symmetric. Without loss of generality, we assume that $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$, where $\lambda_i = \Lambda_{ii}$ are the eigenvalues of A. Since the processes we are concerned with are invariant to scaling A, we also assume that $|\lambda_1| = 1$. We express x as a linear combination of A's orthonormal eigenvectors, $x = c_1u_1 + c_2u_2 + \cdots + c_nu_n$, where u_i are the columns of U. We also have $Ax = c_1\lambda_1u_1 + c_2\lambda_2u_2 + \cdots + c_n\lambda_nu_n$. Similarly, since $A^{-1}U = U\Lambda^{-1}$, we can express the quantization error as $q_{Ax} = c'_1u_1 + c'_2u_2 + \cdots + c'_nu_n$, so

$$A^{-1}q_{Ax} = c'_1\lambda_1^{-1}u_1 + c'_2\lambda_2^{-1}u_2 + \cdots + c'_n\lambda_n^{-1}u_n.$$

The transformation A is useful for quantization when three conditions hold:

(1) For typical inputs x, the norm of Ax is much smaller than the norm of x,

(2) Quantization errors with large $c'_i\lambda_i^{-1}$ for large i (that is, with strong representation for the last eigenvectors) are not disturbing,
(3) $|\lambda_n|$ is not too small.

The first point is important since it implies that $\max_i |(Ax)_i| \ll \max_i |x_i|$, which allows us to achieve a given quantization error with fewer bits. The best choice of norm for this purpose is, of course, the max norm, but algorithmically it is easiler to ensure that $\|Ax\|_2 \ll \|x\|_2$. In particular, this ensures that $\max_i |(Ax)_i| \leq \|Ax\|_2 \ll \|x\|_2 \leq \sqrt{n} \cdot \max_i |x_i|$. Since $\|x\|_2^2 = \sum_i c_i^2$ and $\|Ax\|_2^2 = \sum_i c_i^2\lambda_i^2$, the above condition occurs if and only if the first c_i's are small compared to the last ones. In other words, the first point holds if A, viewed as a filter, filters out strong components of typical x's.

The importance of the second and third points stems from the fact that A^{-1} amplifies the components of q_{Ax} in the direction of the last eigenvectors. If A has tiny eigenvalues, the amplification by a factor λ_i^{-1} is significant for large i. Even if the small eigenvalues of A are not tiny, the error may be unacceptable. The quantization error $A^{-1}q_{Ax}$ always contains moderate components in the direction of eigenvectors that correspond to the small eigenvalues of A. When small error components in these directions distort the signal perceptively, the error will be unacceptable. Therefore, the last two points must hold for the quantization error to be acceptable.

It may seem that the norm of Ax is irrelevant to compression, since one can shrink Ax by a simple scaling, which is clearly useless for compression. The norm of Ax is relevant because we also demand that $|\lambda_1| = \|A\|_2 = 1$. The error is $A^{-1}q_{Ax}$, so $\|A^{-1}q_{Ax}\| \leq \|A^{-1}\|\|q_{Ax}\|$. Making Ax small by scaling A is useless, because it will shrink $\|q_{Ax}\|$ but will expand $\|A^{-1}\|$ by exactly the same factor. But making Ax small while maintaining $\|A\|_2 = 1$ is useful.

5.3.2. Laplacian Transformations. In the following, we discuss the Laplacian matrix of the mesh and its variants and show that these linear transformations work well as quantization transforms.

Let M be a given triangular mesh with n vertices. Each vertex $i \in M$ is conventionally represented using absolute Cartesian coordinates, denoted by $v_i = (x_i, y_i, z_i)$. We denote the *relative* or *δ-coordinates* of v_i as follows:

$$\delta_i = (\delta_i^{(x)}, \delta_i^{(y)}, \delta_i^{(z)}) = d_i v_i - \sum_{k=1}^{d} v_{i_k} \ ,$$

where d_i is the degree of vertex i and i_k is i's kth neighbor. The transformation of the vector of absolute Cartesian coordinates to the vector of relative coordinates can be represented by the matrix $L = D - A$, where A is the mesh adjacency matrix and D is the diagonal matrix $D_{ii} = d_i$.

The matrix L is called the *Laplacian* of the mesh [**34**]. Laplacians of meshes have been extensively studied [**24**], primarily because their algebraic properties are related to the combinatorial properties of the meshes they represent. The Laplacian is symmetric, singular and positive semidefinite. The singularity stems from the fact that the system $Lx = \delta$ has an infinite number of solutions which differ from each other by a vector that is constant on each connected component of the mesh. Thus, we can actually recover x from δ if we know, in addition to δ, the Cartesian coordinate of one x_i in each connected component. We can formalize this method by dropping from L the rows and columns that correspond to one vertex in each connected component, called the *anchor* of the component. The resulting matrix, which we call the *basic invertible Laplacian*, generates all the δ's that we need and is nonsingular. The next section explores other nonsingular variants of the Laplacian.

To explain why variants of the Laplacian are effective quantization transforms, we first have to introduce the notion of mesh frequencies (spectrum). The *frequency* of a real function x defined on the vertices of a mesh M is the number of zero crossings along edges,

$$f(x) = \sum_{(i,j)\in E(M)} \left\{ \begin{array}{ll} 1 & x_i x_j < 0 \\ \\ 0 & \text{otherwise} \end{array} \right\} ,$$

where $E(M)$ is the set of edges of M, so the summation is over adjacent vertices. It turns out that for many classes of graphs, including 3D meshes, eigenvectors of the Laplacian (and related matrices, such as our basic invertible Laplacian) corresponding to large eigenvalues are high-frequency mesh functions, and eigenvectors corresponding to small eigenvalues are low-frequency mesh functions. In other words, when $i \ll j$, $\lambda_i > \lambda_j$ and $f(u_i) \gg f(u_j)$. Furthermore, since 3D models are typically smooth, possibly with some relatively small high-frequency perturbation, the coordinate vectors x, y, and z often have a large low-frequency and a small high-frequency content. That is, the first c_i's are often very small relative to the last ones.

This behavior of the eigenvectors of Laplacians and of typical 3D models implies that the first property we need for effective quantization holds, namely, the 2-norm of Lx is typically much smaller than the norm of x, and therefore the dynamic range of Lx is smaller than that of x. Laplacians also satisfy the second requirement. As stated above, eigenvectors associated with small eigenvalues are low-frequency functions that are typically very smooth. When we add such smooth low-frequency errors to a 3D model, large features of the model may slightly shift, scale, or rotate, but the local

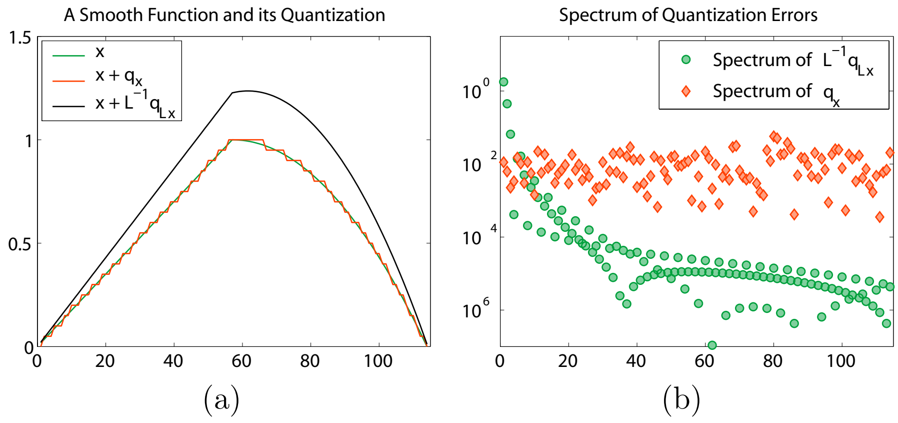


FIGURE 5.3.1. An example of quantization errors in a one-dimensional mesh. The mesh here is a simple chain with 114 vertices (enumerated on the x-axis). (a) shows a smooth function x defined on the mesh, its direct quantization, and a Laplacian-transform quantization. The specific Laplacian that we use here is the 2-anchor invertible Laplacian defined in Section 5.3.3, with anchors at vertices 1 and 114. The quantizations were performed with 20 discrete values uniformly distributed between the minimum and maximum absolute values of the vectors. The direct error vector is smaller in magnitude, but has a strong high-frequency oscillatory nature, whereas the Laplacian-transformed error vector is smooth. (b) explains this observation by plotting, on a log scale, the spectrum of the two errors. We can see that the direct quantization has moderate components in the direction of all eigenvectors of the Laplacian (i.e., all frequencies), whereas the Laplacian-transformed error has strong components in the direction of the smooth eigenvectors, but very small components in the direction of high-frequency eigenvectors.

features and curvature are maintained. Thus, errors consisting mainly of small-eigenvalue low-frequency eigenvectors are not visually disturbing.

However, simple Laplacian transformations do not satisfy our third requirement. The small eigenvalue of a basic invertible Laplacian is typically tiny; a good estimate for $|\lambda_n^{-1}|$ is the product of the maximum topological distance of a vertex from the anchor vertex, and the number of vertices in the mesh (assuming there is one connected component; otherwise the maximum of this estimate over all components) [**16, 50**]. For a typical n-vertex 3D mesh, the small eigenvalue is therefore likely to be $\Theta(n^{-1.5})$. This causes large low-frequency errors which are clearly visible in the example in Figure 5.3.1.

5.3.3. The k-anchor Laplacian. An effective way to increase the small eigenvalue of a Laplacian is to add more anchor points. This section analyzes the effect of two algorithm parameters on the magnitude and shape of the quantization error. One parameter is the number and location of the anchor points. The second parameter is the algorithm that transforms the relative (or δ) coordinates to the original coordinates.

The relationship between the original coordinates x and the relative coordinates δ is given, up to a shift, by the linear system of equations $Lx = \delta$. When we add anchors, we essentially add constraints to this system of equations. Without loss of generality, we assume that the anchors are $x_1, \ldots, x_k$, the first k vertices of the mesh. For each anchor point x_{i_j}, $j = 1, \ldots, k$, we add the constraint $x_i = x_i$, where the left-hand side is taken to be an unknown and the right-hand side a known constant.

It may seem strange that we do not immediately substitute the known constant for the unknown, but the reason for this will become apparent later. The full system of constraints that defines the relationship between the absolute and relative coordinates is therefore

$$\left(\begin{array}{c|c} \multicolumn{2}{c}{L} \\ \hline I_{k\times k} & 0 \end{array}\right) x = \begin{pmatrix} Lx \\ x_{1:k} \end{pmatrix} = \begin{pmatrix} \delta \\ x_{1:k} \end{pmatrix} . \tag{5.3.1}$$

We denote this $(n+k)$-by-n matrix by $\tilde{L}$,

$$\tilde{L} = \left(\begin{array}{c|c} \multicolumn{2}{c}{L} \\ \hline I_{k\times k} & 0 \end{array}\right) , \tag{5.3.2}$$

and call it the *k-anchor rectangular Laplacian.*

With k anchors, the quantized representation of the mesh consists of the quantized δ's and of the absolute coordinates of the anchors. Since we take k to be much smaller than n, there is no need to aggressively quantize the coordinates of the anchors, but they can be quantized as well. The quantized vector that represents the mesh is, therefore,

$$\begin{pmatrix} Lx + q_{Lx} \\ x_{1:k} + q_{x_{1:k}} \end{pmatrix} = \begin{pmatrix} Lx \\ x_{1:k} \end{pmatrix} + q_{\tilde{L}x} = \tilde{L}x + q_{\tilde{L}x} . \tag{5.3.3}$$

The matrix $\tilde{L}$ is rectangular and full rank. Suppose that we try to recover an approximation x' to x from $\tilde{L}x + q_{\tilde{L}x}$. Trying to compute the approximation x' by "solving" the constraint system $\tilde{L}x' = \tilde{L}x + q_{\tilde{L}x}$ for x' will fail, since this system is overdetermined, and therefore most likely inconsistent. An approximation x' can be computed in (at least) two ways. The simplest is to eliminate the last k rows from the system. By adding row $n+j$ to row j, for $j = 1, \ldots, k$ and deleting row $n+j$, we obtain a square symmetric positive definite linear system of equations $\hat{L}x' = b$, which can be solved for x'. This transformation corresponds to multiplying both sides of the system $\tilde{L}x' = \tilde{L}x + q_{\tilde{L}x}$ by an n-by-$(n+k)$ matrix

$$J = \left(\begin{array}{c|c} I_{n\times n} & \begin{array}{c} I_{k\times k} \\ \hline 0 \end{array} \end{array} \right) , \tag{5.3.4}$$

so

$$\hat{L} = J\tilde{L} \tag{5.3.5}$$

and $b = J(\tilde{L}x + q_{\tilde{L}x})$. We call $\hat{L}$ the *k-anchor invertible Laplacian.*[2]

The second method to obtain an approximation x' is to find the least-square solution x' to the full rectangular system $\tilde{L}x' \approx \tilde{L}x + q_{\tilde{L}x}$. It turns out that the norm of the quantization error is essentially the same in the two approximation methods, but the shape of the error is not. The shape of the error when using a least-squares solution to the rectangular system is smoother and more visually pleasing than the shape of the error resulting from the solution of the square invertible system.

5.4. Algebraic Analysis of k-Anchor Laplacians

The norm and shape of the quantization errors in high-pass quantization depend on the spectrum and singular vectors of k-anchor Laplacians. This section presents a detailed analysis of the spectrum of these matrices. In particular, we prove bounds on their smallest and largest singular values. We are mostly interested in the spectrum of the k-anchor rectangular Laplacian, since we can directly relate these to the magnitude of the quantization errors in high-pass quantization.

The section has two parts. The first part, consisting of Subsections 5.4.1–5.4.3, bounds the norm of the error in high-pass quantization. The goal of this part of the section is to prove Lemma 5.4.9 and Theorem 5.4.11. Lemma 5.4.9 shows that the norm of the error is related to the smallest singular value of the rectangular k-anchor Laplacian. How small can this singular value be? Theorem 5.4.11 essentially shows that if every vertex in the graph is reasonably close to an anchor, then this singular value cannot be small. We prove Theorem 5.4.11 by first proving a similar bound on the smallest eigenvalue of the invertible k-anchor Laplacian (in Theorem 5.4.3; this proof is complicated), and then showing how the small singular value of the rectangular Laplacian is related to the small eigenvalue of the invertible Laplacian.

[2]This definition of the k-anchor invertible Laplacian is different than the definition given in [**85**]. The definition that we use here makes the analysis somewhat simpler. The difference is irrelevant to both the algorithms and the analysis, since the k-anchor invertible Laplacian is not used in actual mesh encoding; it is only used as a technical tool in the analysis of the k-anchor rectangular Laplacian.

The second part of the section consists of Subsection 5.4.4, which discusses the shape of the error.

5.4.1. The eigenvalues of $\hat{L}$. In this subsection we show how to bound from below the smallest eigenvalue of $\hat{L}$. Bounding the small eigenvalue from below ensures that the transformation $\hat{L}$ satisfies condition (3) in Section 5.3.1.

The largest eigenvalue $\lambda_{\max}(\tilde{L})$ is at most $2d_{\max}+1$, where $d_{\max}$ is the maximal degree in the mesh [**24**]. This bound is less important than the lower bound on the small eigenvalue, since it only ensures that the norm of the transformed coordinates is never much larger than the norm of the absolute coordinates; in fact, we expect the transformed norm to be much smaller. We include the bound for completeness, and also to show that even when our quantization method is not very effective, it does not cause much harm either.

We first show that bounding the spectrum of $\hat{L}$ proves a lower bound on the quantization error $x - x'$. The bound is similar to the analysis of the quantization error in Section 5.3, but it is not identical. The difference, which turns out to be quite minor, stems from the fact that we now quantize an $(n+k)$-vector, not an n-vector.

LEMMA 5.4.1. *The norm of the quantization error $x - x'$ resulting from solving*

$$\hat{L}x' = J\tilde{L}x' = J(\tilde{L}x + q_{\tilde{L}x})$$

is bounded by $\|x - x'\|_2 \le \sqrt{2}\lambda_{\min}^{-1}(\hat{L})\|q_{\tilde{L}x}\|_2$.

PROOF. We add the quantization error $q_{\tilde{L}x}$ to the right-hand side of Equation 5.3.1, and multiply both sides by J,

$$J\tilde{L}x' = J(\tilde{L}x + q_{\tilde{L}x}) \ . \tag{5.4.1}$$

Because $J\tilde{L}x' = \hat{L}x'$, we can multiply both sides by $\hat{L}^{-1}$ to obtain

$$x' = \hat{L}^{-1}J(\tilde{L}x + q_{\tilde{L}x}) = \hat{L}^{-1}(\hat{L}x + Jq_{\tilde{L}x}) \quad = \quad x + \hat{L}^{-1}Jq_{\tilde{L}x} \ ,$$

so

$$\|x - x'\|_2 \le \|\hat{L}^{-1}\|_2\|J\|_2\|q_{\tilde{L}x}\|_2 \ .$$

We now bound the first two factors in the right-hand-side product. Because $\hat{L}$ is symmetric positive definite,

$$\|\hat{L}^{-1}\|_2 = \lambda_{\max}(\hat{L}^{-1}) = 1/\lambda_{\min}(\hat{L}) = \lambda_{\min}^{-1}(\hat{L}) \ .$$

By the definition of the 1 and infinity norms,

$$\|J\|_2^2 \le \|J\|_1\|J\|_\infty = 1 \cdot 2 = 2 \ ,$$

which completes the proof. □

This lemma shows that to preserve the bound on the norm of $x - x'$, the quantization error $q_{x_{1:k}}$ for the anchor points should be no larger than the quantization error q_{Lx} of the relative coordinates.

We now bound the smallest eigenvalue of $\hat{L}$. We express the lower bound in terms of a set of paths in the mesh. Given a set of anchor points, we assign each vertex a path to an anchor point. The bound uses the following three metrics of the set of paths.

Definition 5.4.2. The *dilation* ϑ of the set of paths is the length, in edges, of the longest path in the set. The *congestion* φ of the set is the maximal number of paths that use a single edge in the mesh. The *contention* ϱ of the set is the maximal number of vertices whose paths lead to a single anchor point. The maximum is taken over all vertices for dilation, over all edges for congestion, and over all anchors for contention.

The smaller the dilation, congestion, and contention, the better the bound on the small eigenvalue of $\hat{L}$. Note that for a single set of anchor points, we can assign many different sets of paths, some of which yield tighter bounds than others. In addition, even the best set of paths does not, in general, provide a completely tight bound. For more details, see [**16**]. But the dependence of the bound on the dilation, congestion and contention does provide us with guidelines as to how to select the anchor points. The next theorem is the main result of this subsection.

Theorem 5.4.3. *The smallest eigenvalue of $\hat{L}$ satisfies*

$$\lambda_{\min}(\hat{L}) \geq \frac{1}{\varphi \cdot \vartheta + \varrho} .$$

We use the following strategy to prove this theorem. We will show how to factor $\hat{L}$ into $\hat{L} = VV^T$. The eigenvalues of $\hat{L}$ are the squares of the singular values of V, so it suffices to bound the small singular value of V. The factor V will have a special structure, in which each column corresponds to one edge of the mesh or to one anchor point. We will then use the given set of paths from vertices to anchor points, to construct a matrix W such that $VW = I$, and show how the norm of W is related to the path structure. The equation $VW = I$ will allow us to relate the 2-norm of W, which we can bound using the path set, to the small singular value of V, which we seek to bound.

The following definitions are used in the construction of the factor V. For convenience, we repeat the definitions of *edge-vectors* and *vertex-vectors* from Section 3.2.

Definition 5.4.4. The *edge-vector* $\langle ij \rangle$ in $\mathbb{R}^n$ is a vector with exactly two non-zeros, $\langle ij \rangle_{\min(i,j)} = 1$ and $\langle ij \rangle_{\max(i,j)} = -1$. The *vertex-vector* $\langle i \rangle$ in $\mathbb{R}^n$ is a vector with exactly one non-zero, $\langle i \rangle_i = 1$.

We associate an edge-vector $\langle ij \rangle$ with an edge connecting vertex i with vertex j. The following lemma demonstrates one of the connection between edges and their corresponding vectors.

LEMMA 5.4.5. *The edge-vectors of a simple path between vertices i and j span the edge-vector $\langle ij \rangle$ with coefficients ± 1.*

The following lemma describes a factorization of k-anchor Laplacian matrices:

LEMMA 5.4.6. *A k-anchor Laplacian matrix $\hat{L}$ can be factored into $\hat{L} = VV^T$, such that $V = \begin{pmatrix} V_1 & V_2 \end{pmatrix}$, where V_2 is a matrix of unscaled edge-vectors, each column corresponding to one non-zero off-diagonal in $\hat{L}$, and V_1 is a matrix of vertex-vectors, each column corresponding to an anchor point.*

PROOF. For each off-diagonal nonzero $\hat{l}_{ij} = -1$ (each edge of the mesh), V has a column containing the edge vector $\langle ij \rangle$, and for each anchor j, V has a vertex vector $\langle j \rangle$. The edge vectors constitute V_2 and the vertex vectors constitute V_1. It is easy to verify that $\hat{L} = VV^T$. For a more detailed proof, see [**13**]. $\square$

Given the above factorization, we bound the smallest singular value of V. Our course of action in bounding the smallest singular value of V is as follows: we shall find a matrix $W \in R^{m \times n}$ such that $VW = I_{n \times n}$. As the next lemma shows, the matrix G with the smallest 2-norm satisfying $VG = I_{n \times n}$ is the Moore-Penrose pseudo-inverse $G = V^+$ of V [**40**, pages 257–258]. Therefore, any matrix W satisfying $VW = I_{n \times n}$ has the property $\|W\| \geq \|V^+\|$. We shall then find an upper bound C on $\|W\|$. Since $C \geq \|W\| \geq \|V^+\| = \frac{1}{\sigma_{\min}(V)}$ we will be able to conclude that $\sigma_{\min}(V) \geq \frac{1}{C}$. We first prove a technical lemma concerning the pseudo-inverse (this result is probably well-known, but we have not found it in the literature).

LEMMA 5.4.7. *Let V be a full-rank n-by-m real matrix, and let G be an m-by-n real matrix such that $VG = I_{n \times n}$. Then $\|G\|_2 \geq \|V^+\|_2$.*

PROOF. The singular values of V^+V are n ones and $m - n$ zeros, so its 2-norm is 1. We now show that for any x with unit 2-norm we have $\|V^+x\|_2 \leq \|Gx\|_2$. Let $c = \|Gx\|_2$, and let $y = Gx/c$, so $\|y\|_2 = 1$. We have $Gx = cy$, and multiplying V from the left on both sides we get $x = Ix = VGx = cVy$. Multiplying now from the left by V^+ we get $V^+x = cV^+Vy$, so $\|V^+x\|_2 = \|cV^+Vy\|_2 \leq c\,\|V^+Vy\|_2 \leq c\,\|V^+V\|_2\,\|y\|_2 = c \cdot 1 \cdot 1 = c = \|Gx\|_2$. $\square$

We are now ready to bound the singular values of V.

LEMMA 5.4.8. *Given a k-anchor Laplacian $\hat{L}$ with a factorization into edge and vertex vectors $\hat{L} = VV^T$ as in Lemma 5.4.6, and a set of paths*

$$\Pi = \{\pi_i = (i, i_1, i_2, \ldots, j) | i = 1, \ldots, n \text{ and } j \text{ is an anchor}\} \ ,$$

we have

$$\sigma_{\min}(V) \geq \frac{1}{\sqrt{\varphi(\Pi) \cdot \vartheta(\Pi) + \varrho(\Pi)}} \ .$$

PROOF. Finding a matrix W satisfying $VW = I_{n \times n}$ is equivalent to finding a vector w_i, for $i = 1, \ldots, n$, such that $Vw_i = e_i$, where $e_i = \langle i \rangle$ is the ith unit vector.

Let j_i be the anchor endpoint of π_i. It is easy to verify that

$$\langle ij_i \rangle = (-1)^{(i>j_i)} \sum_{(\ell_1,\ell_2) \in \pi_i} (-1)^{(\ell_1 > \ell_2)} \langle \ell_1 \ell_2 \rangle \ .$$

(We use the convention that a boolean predicate such as $(i > j)$ evaluates to 1 if it is true and to 0 otherwise.) By Lemma 5.4.6, all the edge vectors in the summation are columns of V. To obtain w_i, it remains is to add or subtract $\langle j_i \rangle$, and perhaps to multiply by -1,

$$\langle i \rangle = (-1)^{(i>j_i)} \langle ij_i \rangle + \langle j_i \rangle \ .$$

The last two equations together specify w_i, which contains only 1's, -1's, and 0's.

Now that we have found, column by column, a matrix W such that $VW = I_{n \times n}$, we partition the rows of W such that

$$VW = (V_1 V_2) \begin{pmatrix} W_1 \\ W_2 \end{pmatrix} \ .$$

The rows of W_1 correspond to the columns of V_1, the vertex vectors in V, and the rows of W_2 corresponds to the columns of V_2, the edge vectors in V. We will bound the norm of W by bounding separately the norms of W_1 and of W_2. We first bound $\|W_1\|_2$:

$$\begin{aligned}
\|W_1\|_2^2 &\leq \|W_1\|_1 \|W_1\|_\infty \\
&= \left(\max_j \sum_i |[W_1]_{ij}| \right) \left(\max_i \sum_j |[W_1]_{ij}| \right) \\
&= 1 \cdot \varrho(\Pi) \ .
\end{aligned}$$

The 1-norm of W_1 is one since there is exactly one nonzero in each column of i, in position j_i, and its value is 1. The ∞-norm of W_1 is the contention of the path set, since each row of W_1 corresponds to one anchor point, and it appears with value 1 in each path (column) that ends in it. Therefore, each row in W_1 contains at most $\varrho(\Pi)$ 1's, and the other entries are all 0.

Bounding $\|W_2\|_2$ is similar. Each row in W_2 corresponds to one edge of the mesh and each column to a path in Π. Each edge is used in at most $\varphi(\Pi)$ paths, so $\|W_2\|_\infty = \varphi(\Pi)$. Each path contains at most $\vartheta(\Pi)$ edges, so $\|W_2\|_1 = \vartheta(\Pi)$. Therefore,

$$\begin{aligned}
\|W\|_2^2 &= \max_{\|x\|_2=1} \|Wx\|_2^2 \\
&= \max_{\|x\|_2=1} \left\| \begin{matrix} W_1x \\ W_2x \end{matrix} \right\|_2^2 \\
&= \max_{\|x\|_2=1} \left(\|W_1x\|_2^2 + \|W_2x\|_2^2\right) \\
&\leq \max_{\|x_1\|_2=1} \|W_1x_1\|_2^2 + \max_{\|x_2\|_2=1} \|W_2x_2\|_2^2 \\
&= \|W_1\|_2^2 + \|W_2\|_2^2 \\
&\leq \varphi(\Pi)\,\vartheta(\Pi) + \varrho(\Pi)\ .
\end{aligned}$$

The bound on $\sigma_{\min(V)}$ follows immediately from the bound on $\|W\|_2$ and from the discussion preceding the statement of the lemma. $\square$

Now we can conclude that $\lambda_{\min}(\hat{L}) \geq \frac{1}{\varphi\cdot\vartheta+\varrho}$. This follows from two facts: (1) in a symmetric positive definite matrix the singular values are the same as the eigenvectors, therefore $\lambda_{\min}(\hat{L}) = \sigma_{\min}(\hat{L})$. (2) if $\hat{L} = VV^T$ then the singular values of $\hat{L}$ are the squares of the singular values of V (this follows directly from V's SVD decomposition).

We can now easily prove Theorem 5.4.3:

Proof. $\lambda_{\min}(\hat{L}) = \sigma_{\min}(\hat{L}) = \sigma^2_{\min}(V) \geq \frac{1}{\varphi\cdot\vartheta+\varrho}$. $\square$

5.4.2. Bounding the quantization error using the singular values of $\tilde{L}$.

We now show that if we define x' as the least-squares minimizer of $\|\tilde{L}x' - \tilde{L}x + q_{\tilde{L}x}\|_2$, the norm of the error $x - x'$ can be bounded using estimates on the singular values of $\tilde{L}$. The analysis below is equivalent to the analysis in Lemma 5.4.1, but for the case of $\tilde{L}$, the k-anchor rectangular Laplacian rather than for the case of $\hat{L}$, the square k-anchor invertible Laplacian.

Lemma 5.4.9. *Let x' be the least-squares minimizer of $\|\tilde{L}x'-\tilde{L}x+q_{\tilde{L}x}\|_2$. The norm of the error $x - x'$ is bounded by*

$$\|x - x'\|_2 \leq \sigma^{-1}_{\min}(\tilde{L})\|q_{\tilde{L}x}\|_2\ ,$$

where $\sigma_{\min}(\tilde{L})$ denotes the nth and smallest singular value of $\tilde{L}$.

PROOF. We express x' in terms of the Moore-Penrose pseudo-inverse $\tilde{L}^+$ of $\tilde{L}$,

$$x' = \tilde{L}^+ \left(\tilde{L}x + q_{\tilde{L}x} \right) = x + \tilde{L}^+ q_{\tilde{L}x} \ .$$

Therefore,

$$\|x - x'\|_2 \leq \|\tilde{L}^+\|_2 \|q_{\tilde{L}x}\|_2 = \sigma_{\min}^{-1}(\tilde{L}) \|q_{\tilde{L}x}\|_2 \ .$$

□

5.4.3. The singular values of $\tilde{L}$. The next step is to show that the singular values of $\tilde{L}$ cannot be much smaller than the smallest eigenvalue of $\hat{L}$. In fact, we show that they are at most a factor of $\sqrt{2}$ smaller.

The proof of Lemma 5.4.1 shows that the 2-norm of J is at most $\sqrt{2}$. It is easy to show that the norm is, in fact, exactly $\sqrt{2}$, and that all the singular values of J are either 1 or $\sqrt{2}$. The next lemma shows that the $\sqrt{2}$ bound on the norm of J ensures that $\sigma_{\min}(\tilde{L}) \geq \lambda_{\min}(\hat{L})/\sqrt{2}$.

LEMMA 5.4.10. *Let A, B, and C be matrices such that $AB = C$. Then*

$$\sigma_{\min}(B) \geq \frac{\sigma_{\min}(C)}{\sigma_{\max}(A)} \ .$$

PROOF. Suppose for contradiction that

$$\sigma_{\min}(B) = \epsilon < \sigma_{\min}(C)/\sigma_{\max}(A) \ .$$

Then there exist vectors x and y such that $\|x\|_2 = \|y\|_2 = 1$ and $Bx = \epsilon y$. (x and y are the right and left singular vectors corresponding to $\sigma_{\min}(B)$.) Therefore,

$$\begin{aligned} \|Cx\|_2 &= \|ABx\|_2 = \|A\epsilon y\|_2 = \epsilon \|Ay\|_2 \leq \\ &\leq \epsilon \sigma_{\max}(A) \|y\|_2 = \epsilon \sigma_{\max}(A) < \sigma_{\min}(C) \ , \end{aligned}$$

a contradiction. □

We can now prove the main theorem of this subsection.

THEOREM 5.4.11.

$$\sigma_{\min}(\tilde{L}) \geq \frac{\lambda_{\min}(\hat{L})}{\sqrt{2}} \ .$$

PROOF. Since $J\tilde{L} = \hat{L}$, by the previous lemma

$$\sigma_{\min}(\tilde{L}) \geq \frac{\sigma_{\min}(\hat{L})}{\sigma_{\max}(J)} = \frac{\sigma_{\min}(\hat{L})}{\sqrt{2}} = \frac{\lambda_{\min}(\hat{L})}{\sqrt{2}} \ .$$

□

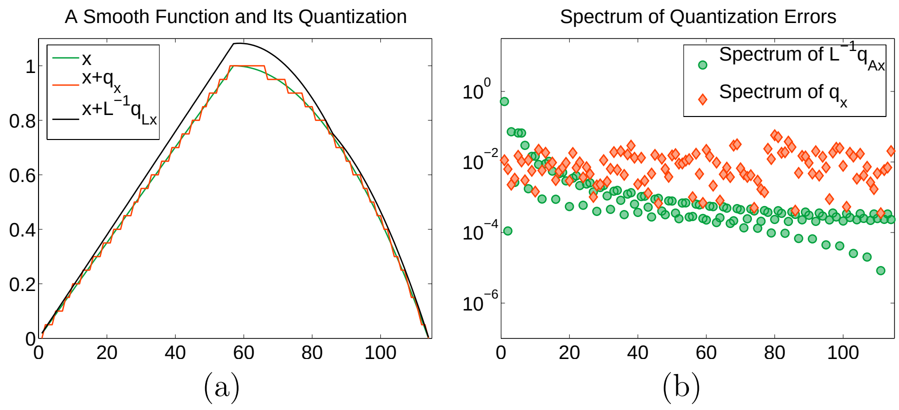


FIGURE 5.4.1. The same mesh as in Figure 5.3.1, but with an additional anchor point at vertex 86. The Laplacian here is a 3-anchor *invertible* Laplacian with anchors at vertices 1, 86, and 114. The transformed quantization error is *not* smooth at the anchor point, even though the vector x is smooth there.

5.4.4. Singular vectors and the shape of the error. Why do we propose to use a rectangular Laplacian rather than a square invertible one? The reason lies in the shape of the quantization error that each method generates. We have already seen that adding anchor points increases the smallest singular value of both the invertible and the rectangular Laplacians. Furthermore, in both cases the 2-norm of the error $x - x'$ is bounded by $\sqrt{2}\lambda_{\min}^{-1}(\hat{L})\,\|q_{\tilde{L}x}\|_2$, exactly the same bound. (The actual errors will differ and the norms will most likely differ, since the bounds are not tight, but the bounds we proved are exactly the same.) We have found, however, that the shape of the error is visually better when we obtain the approximation x' from the rectangular Laplacian. The main difference between the two errors is that the rectangular approximation x' is usually smooth where x is smooth, but the invertible approximation is not. The invertible approximation is almost always non-smooth at the anchors, where “spikes” seem to always appear.

The crucial observation is that the k-anchor invertible Laplacian essentially forces the error $x - x'$ to zero at the anchors, and allows the error to grow as we get farther and farther away from the anchor points. When we obtain x' from solving a least-squares problem whose coefficient matrix is $\tilde{L}$, x' can differ from x everywhere, including at the anchor points. This allows x' to be smooth.

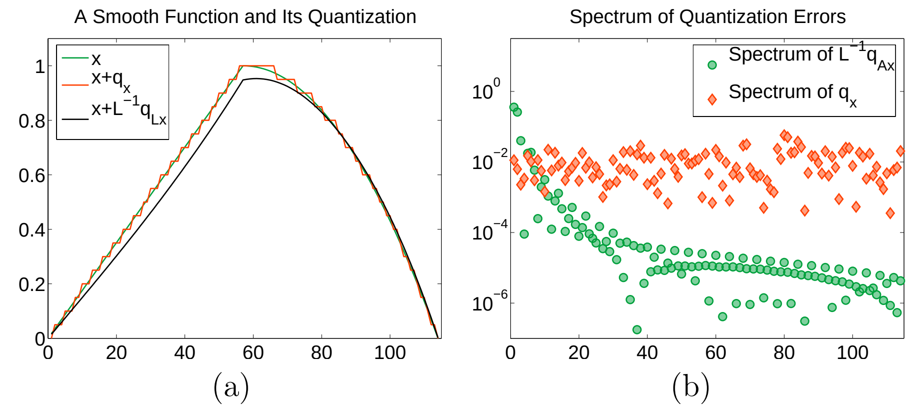


FIGURE 5.4.2. The same mesh as in Figures 5.3.1 and 5.4.1. The Laplacian here is a 3-anchor *rectangular* Laplacian with anchors at vertices 1, 86, and 114. The transformed quantization error *is* smooth at the anchor point.

Formalizing this explanation is hard and is beyond the scope of this chapter. The error $x' - x$ consists, in both cases, mainly of the singular vectors of $\tilde{L}$ or $\hat{L}$ that correspond to the smallest singular values. If these singular vectors are smooth, the error $x - x'$ will be smooth, so x' will be smooth where x is smooth. Are these vectors smooth? The numerical example in Figure 5.4.1 indicates that the relevant singular/eigen vectors of $\hat{L}$ are *not* smooth. The numerical example in Figure 5.4.2 indicates that the singular vectors of $\tilde{L}$ that correspond to small singular values *are* smooth.

In this chapter we do not attempt to prove these statements about the shape of the singular vectors. In general, the singular *vectors* of Laplacian and Laplacian-like matrices have not been researched as much as the singular *values*. It is generally believed that the vectors corresponding to small singular values are indeed smooth. This belief underlies important algorithms such as multigrid [**20**] and spectral separators [**76**]. Some additional progress towards an understanding of the relationships between the graph and the eigenvectors of its Laplacian were made recently by Ben-Chen and Gotsman [**7**]. On the other hand, there is also research that indicates that these vectors are not always well-behaved [**50**].

We leave the full mathematical analysis of the shape of the errors as an open problem in this chapter; the empirical evidence shown in [**85**] supports our claim.

5.5. The Effect of Anchor Points on Numerical Accuracy

So far we have analyzed the norm of the error assuming that x' is the exact solution of $\hat{L}x' = J(\tilde{L}x + q_{\tilde{L}x})$ or exact minimizer of $\left\|\tilde{L}x' - (\tilde{L}x + q_{\tilde{L}x})\right\|_2$. Since we cannot determine x' exactly using floating-point arithmetic, what we actually obtain is an approximation x'' to x'. The total error $x - x''$ depends on both $x - x'$ and $x' - x''$. In this section we analyze the numerical error $x' - x''$, and show that it too depends primarily on the small singular values of the coefficient matrices $\hat{L}$ and $\tilde{L}$, and hence on the anchor points. The results in this section rely on standard error bounds from numerical linear algebra. For details on these error bounds, see for example [**59**] or [**92**]; the first reference is an encyclopedic monograph, the second a readable textbook.

We assume that the approximation x'' is obtained using a *backward stable* algorithm. For the invertible problem, this means that x'' is the exact solution of $(\hat{L} + \delta\hat{L})x'' = J(\tilde{L}x + q_{\tilde{L}x})$, where $\delta\hat{L}$ is a small perturbation such that $\|\delta\hat{L}\|/\|\hat{L}\| = O(\epsilon_{\text{machine}})$, where $\epsilon_{\text{machine}}$ is a small constant depending on the floating-point arithmetic, about 10^{-16} for double-precision IEEE-754 arithmetic, which is now used on virtually all computers. For the rectangular problem, backward stability means that x'' is the exact minimizer of $(\tilde{L} + \delta\tilde{L})x'' - (\tilde{L}x + q_{\tilde{L}x})$ for a similarly small perturbation.

Since $\hat{L}$ is a symmetric positive-definite matrix and since $\tilde{L}$ is full rank, most linear-equation solvers and most least-squares solvers are backward stable when applied to them. This includes sparse direct Cholesky factorization solvers for the square problem, sparse QR solvers for the rectangular least-squares problem, and most iterative algorithms for these problems.

When we obtain an approximation x'' using a backward stable algorithm, the relative norm of the so-called forward error $x' - x''$ is bounded by the *condition number* κ of the problem times $\epsilon_{\text{machine}}$,

$$\frac{\|x'' - x'\|}{\|x'\|} = O(\kappa\epsilon_{\text{machine}}) \ . \tag{5.5.1}$$

For the invertible problem, the condition number is simply the condition number of the coefficient matrix,

$$\kappa_{\text{inv}} = \left\|\hat{L}\right\| \left\|\hat{L}^{-1}\right\| \ . \tag{5.5.2}$$

The norm in Equation (5.5.2) is the matrix norm induced by the vector norm in Equation (5.5.1). When we use the 2-norm in Equations (5.5.2) and (5.5.1), we have

$$\kappa_{\text{inv}} = \frac{\sigma_{\max}(\hat{L})}{\sigma_{\min}(\hat{L})} \ .$$

The quantity $\sigma_{\max}(\hat{L})/\sigma_{\min}(\hat{L})$ is called the *spectral condition number* of $\hat{L}$ and is denoted by $\kappa_2(\hat{L})$ or simply κ_2 when the matrix is clear from the context.

The condition number of least-squares problems is a little more complicated. We denote by θ the angle between the right-hand side $(\tilde{L}x+q)$ (here $q = q_{\tilde{L}x}$) and its projection into the column space of $\tilde{L}$. Since $\tilde{L}x$ is in this column space, the size of $\tan\theta$ is roughly proportional to $\|q\|_2/\|\tilde{L}x\|$, which is proportional to how aggressive the quantization is. Therefore, $\tan\theta$ will be usually small. We denote by η the quantity

$$\eta = \frac{\|\tilde{L}\|_2\|x\|_2}{\|\tilde{L}x\|_2} \ .$$

This quantity is bounded by $1 \le \eta \le \kappa_2(\tilde{L})$. In our case, unfortunately, η will not be large, because $\tilde{L}x$ contains some values of x, namely the anchors, so its norm will not be much smaller than the norm of x. Given θ and η, we can express the condition number of solving least squares problems,

$$\kappa_{\text{rect}} = \kappa_2(\tilde{L}) + \frac{\kappa_2(\tilde{L})^2\tan\theta}{\eta} \ .$$

In our case, $\kappa_2(\tilde{L})$ and $\kappa_2(\hat{L})$ depend only on the small eigenvalue of $\hat{L}$, which we have already shown to be strongly influenced by the anchor points. Since $\sigma_{\max}(\hat{L}) = \lambda_{\max}(\hat{L}) \le 2d_{\max} + 1$, where $d_{\max}$ is the maximal degree of a vertex in the mesh, and since $\sigma_{\max}(\tilde{L}) \le \sqrt{2}\lambda_{\max}(\hat{L})$, in both cases the largest singular value is bounded by a small constant, so $\kappa(L) = O(\lambda_{\min}^{-1}(\hat{L}))$ for both L's.

THEOREM 5.5.1. *Let* $\lambda = \lambda_{\min}(\hat{L})$*,* $\epsilon = \epsilon_{\text{machine}}$*, and* $q = q_{\tilde{L}x}$*. The 2-norm of the error* $x - x''$*, when* x'' *is computed from the invertible Laplacian using a backward-stable algorithm, is bounded by*

$$\|x - x''\|_2 \le O(\lambda^{-1}\|q\|_2 + \lambda^{-1}\epsilon\|x\|_2 + \lambda^{-2}\epsilon\|q\|_2) \ .$$

PROOF. By Lemma 5.4.9, $\|x'\|_2 = \|x' + x - x\|_2 \le \|x - x'\|_2 + \|x\|_2 \le \lambda^{-1}\|q\|_2 + \|x\|_2$. The inequality and the discussion preceding the theorem yield

$$\begin{aligned}
\|x - x''\|_2 &= \|x - x' + x' - x''\|_2 \\
&\le \|x - x'\| + \|x' - x''\|_2 \\
&\le \lambda^{-1}\|q\|_2 + \|x' - x''\|_2 \quad \text{by Lemma 5.4.9} \\
&\le \lambda^{-1}\|q\|_2 + O(\kappa_2(\hat{L})\epsilon\|x'\|_2) \\
&= \lambda^{-1}\|q\|_2 + O(\lambda^{-1}\epsilon(\lambda^{-1}\|q\|_2 + \|x\|_2)) \\
&= \lambda^{-1}\|q\|_2 + O(\lambda^{-1}\epsilon\|x\|_2 + \lambda^{-2}\epsilon\|q\|_2) \,.
\end{aligned}$$

□

We now state the corresponding theorem for the least squares case. The proof, which we omit, is identical except for the expression of the condition number.

THEOREM 5.5.2. *Let $\lambda = \lambda_{\min}(\hat{L})$, $\epsilon = \epsilon_{\text{machine}}$, and $q = q_{\tilde{L}x}$. The 2-norm of the error $x - x''$, when x'' is computed from the rectangular least-squares problem using a backward-stable algorithm, is bounded by*

$$\|x - x''\|_2 \leq O\left(\lambda^{-1}\|q\|_2 + \lambda^{-1}\epsilon\|x\|_2 + \tfrac{\lambda^{-2}\tan\theta\epsilon\|x\|_2}{\eta} + \lambda^{-2}\epsilon\|q\|_2 + \tfrac{\lambda^{-3}\tan\theta\epsilon\|q\|_2}{\eta}\right) .$$

$$\begin{aligned}\|x - x''\|_2 \leq\ & O\left(\lambda^{-1}\|q\|_2 + \lambda^{-1}\epsilon\|x\|_2 + \frac{\lambda^{-2}\tan\theta\epsilon\|x\|_2}{\eta}\right.\\ & \left. +\lambda^{-2}\epsilon\|q\|_2 + \frac{\lambda^{-3}\tan\theta\epsilon\|q\|_2}{\eta}\right)\end{aligned}$$

One way of solving the least-squares problem is by constructing and solving the so-called *normal equations.* This solution method relies on the fact that the least-squares minimizer x' is also the solution of the symmetric positive-definite linear system $\tilde{L}^T\tilde{L}x' = \tilde{L}^T(\tilde{L}x + q_{\tilde{L}x})$. Even when the normal equations are solved using a backward-stable algorithm, the whole algorithm is not backward stable with respect to the original least-squares problem. The computed solution satisfies only

$$\frac{\|x'' - x'\|_2}{\|x'\|_2} = O\left(\kappa_2(\hat{L})^2\epsilon_{\text{machine}}\right) .$$

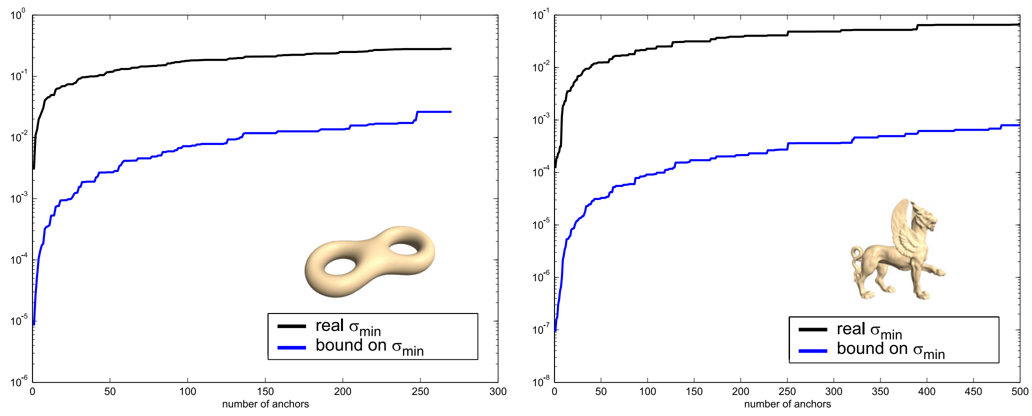
Because the error bound is much larger in this case (and usually much larger in practice), this method is usually not recommended. However, since in our application we can control and estimate $\kappa_2(\hat{L})$ by adding anchor points, we can ensure that even the normal-equations forward error is acceptable.

5.6. Algorithmic issues and results

Two algorithmic problems arise in the high-pass quantization method: the anchor-selection problem, and the linear least-squares problem. This section explains how these issues can be addressed and shows some experimental results.

5.6.1. Evaluating the bound on σ_{min}.

In order to exploit the theoretical results presented in the previous section, we need an algorithm to evaluate the lower bound on σ_{min}, the smallest singular value of $\tilde{L}$. Given a set of anchor vertices $\{a_1, a_2, ..., a_k\}$, we are looking for some partition of all the mesh vertices into k subsets, such that we can define the values $\varphi, \vartheta, \varrho$ (congestion, dilation and contention) reasonably. Since finding a partition



Eight mesh, 2718 vertices   *Feline* mesh, 49864 vertices

FIGURE 5.6.1. Comparison of the congestion-dilation-contention bound on σ_{min} (see Theorem 5.4.11) with the actual value of σ_{min}. The x-axis shows the number of anchors used.

that strictly maximizes the bound in Theorem 5.4.11 does not seem feasible, we use the following heuristic. We simultaneously grow patches of vertices around the anchors by running k-source BFS. This algorithm produces a rather balanced partition that keeps the values of ϑ and ϱ small. After the partition has been computed, the calculation of ϑ and ϱ is straightforward. To compute φ, we use the parent pointers stored for each vertex during the BFS procedure. These pointers define the tree of paths from each vertex to the root (source anchor vertex). Clearly, the most "loaded" edges are the edges whose source vertex is the root. By counting the number of vertices in the subtrees hanging on those edges, we obtain their edge loads, and compute the maximum over all the k subsets.

We have compared the evaluation of the lower bound of σ_{min} with the real value of σ_{min} on moderately-sized meshes. The accurate value of σ_{min} was computed in MATLAB. Figure 5.6.1 shows two representative graphs summarizing this experiment. The horizontal axis in the graphs represents the number of anchor rows present in $\tilde{L}$. We incrementally added random anchor vertices and plotted the value of σ_{min} and the lower bound. As can be seen from these graphs, the bound differs from the real value by 1.5–2.5 orders of magnitude and behaves consistently with the real σ_{min}. We can thus conclude that our theoretical bound is not too pessimistic and can be used in practical algorithms for choosing the anchors, as discussed below.

5.6.2. Algorithms for placing anchor points. Sorkine et al. use the following adaptive and greedy algorithm to select anchor points. They begin by placing one random anchor point and generating a 1-anchor rectangular Laplacian, denoted by $\tilde{L}_1$. They then use this matrix to transform the coordinates, quantize the δ-coordinates, compute an approximation x''_1, and compute the error $x - x''_1$. The second anchor is placed at the vertex with the largest error, to yield $\tilde{L}_2$. These iterations continue either until a satisfactory error is attained, or until a given number k of anchors is placed.

The advantage of this scheme is that it directly attempts to minimize the reconstruction error, rather than its bound. However, the first iterations of the greedy algorithm may compute ineffective anchors, since in the beginning, only a few anchors are used, and the matrix $\tilde{L}$ is thus ill-conditioned. Therefore, the first reconstructed vectors x''_i will contain very high errors.

The congestion-dilation-contention bounds that we present in this chapter suggest another anchor-selection scheme, one that aims to maximize the lower bound on σ_{min}. This scheme can be used to select enough effective anchors to ensure reasonable conditioning of $\tilde{L}$, and more anchors can then be added using the previous greedy algorithm.

As mentioned above, it is hard to strictly maximize the bound on σ_{min}. To choose anchors so as to make the bound expression larger, we again propose a heuristic method. It selects the anchors one by one while minimizing the value of ϑ. The method operates as follows. We start with one randomly chosen anchor and compute its edge-distance from all the other vertices in the mesh by running BFS. The furthest vertex is chosen as the next anchor, and we proceed in the same manner. In the i-th iteration, we have a set of i anchor vertices; we run i-source BFS from these vertices to find the vertex that achieves the longest edge distance from an anchor (the value of ϑ). This vertex is assigned as the $(i+1)$-th anchor. The procedure stops when we reach a large enough value of the bound or after a prescribed number of steps. It should be noted that actually there is no need to run the complete i-source BFS in every step. It is enough to run (partial) BFS from the last chosen anchor in order to update the distances. The front propagation of the BFS procedure stops whenever we meet a vertex whose old distance value is smaller than the distance that would be assigned by the current BFS.

Figure 5.6.2 shows some steps of the above anchor-selection algorithm on the *Feline* model. The anchors are well-spaced, which is favorable for the congestion-dilation-contention bound. The graph in Figure 5.6.2 plots the reconstruction max-norm as function of the number of anchors. The red line denotes the values for anchors chosen with the bound-maximizing scheme, while the blue line represents the greedy scheme used in [**85**]. As expected, the greedy scheme produces somewhat smaller errors since it operates directly to minimize the max-norm error. However, on larger

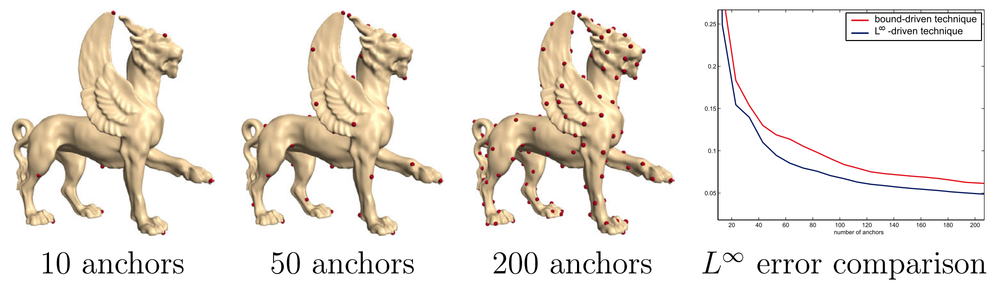


FIGURE 5.6.2. Anchors on the *Feline* model chosen by the bound-driven algorithm. The first three images display stages of the incremental anchor selection procedure. The even spacing of the anchors increases the lower bound on σ_{min}. The graph compares the L^∞ reconstruction error for anchors chosen with the greedy scheme as in **[85]** (in red) and the bound-driven scheme (in blue), when using the same level of δ-coordinates quantization (6 bits/coordinate).

meshes, the error bound minimization scheme gives an initial set of anchors to make $\tilde{L}$ well-conditioned and thus provides a good starting point for the greedy algorithm. Moreover, this scheme is much faster since it does not require reconstruction of the Cartesian coordinates at each iteration.

After the anchor placement strategy has been fixed, the shape of the mesh reconstructed with the high-pass quantization technique depends mainly on two factors: the level of δ-coordinates quantization and the amount of anchors. The Cartesian coordinates of the anchors should be mildly quantized to preserve accuracy. Adding anchors to the representation is "cheap": if B is the number of bits per coordinate (typically, between 12–14), then a single anchor requires $3B + \log(n)$ bits ($\log(n)$ bits for the index of the anchor vertex). As suggested by the theoretical bounds in Theorems 5.4.3 and 5.4.11, we can keep the condition number of the system (and hence the L^2 error) constant by ensuring that the dilation, congestion, and contention are bounded by a constant. When the number of anchors is a constant fraction p of the number of mesh vertices n, the dilation, congestion, and contention are usually bounded by a constant or grow very slowly with n. As discussed in **[85]**, for visually acceptable value of L^2 error, p is rather small, up to 1%. This is due to the fact that the visual quality is more affected by the change of high-frequency details (e.g. surface normals or the surface local smoothness properties), rather than global low-frequency errors. Adding anchors to a fixed δ-quantization only helps to make the low-frequency error smaller, but almost does not effect the high-frequency error. On the other hand, adding more bit planes to the δ-coordinates significantly reduces the high-frequency error, as well as the low-frequency error (see the $\|q_{\tilde{L}x}\|_2$ component of the L^2 error bound in Lemma 5.4.9). However, this is more expensive since adding a single bit per δ-coordinate requires addition of n bits to the representation (prior to entropy-coding).

The visual tables in Figures 5.7.1 and 5.7.2 demonstrate the effect of adding anchors versus adding bits to the δ-coordinates. Each row in the tables displays reconstructed models with varying number of anchors, for a fixed δ-quantization level. As can be seen in the figures, the surface smoothness properties vary in different rows, but not columns, while the surface general "pose" (affected by low-frequency error) decreases both in the rows (top to bottom, as more bits are added to the δ-coordinates) and in the columns (left to right, as more anchors are added). This is also supported numerically by the values of the S_q and M_q errors (see [**85**]). In all the experiments, fixed quantization of 12 bits/coordinate was applied to the positions of the anchor vertices. The file sizes given below each reconstruction were obtained by arithmetic encoding of the quantized δ-coordinates and the anchors.

It is important to note that state-of-the-art geometry encoding methods, such as the wavelet compression [**66**], employ zerotree encoding with a clever bit allocation scheme that adapts to the local surface shape. We believe that adaptive encoding will benefit our geometry encoding scheme as well; currently, we uniformly quantize the δ-coordinates of the entire mesh and encode them with a standard arithmetic encoder, which does not fully exploit the specific nature of the data. This rather naive compression is obviously not optimal, as supported by the statistics in Table 5.6.3, where we compare the file sizes of the models compressed by our method with those of [**66**]. However, in contrast to [**52, 66**] and others, our method does not require any remeshing. It would be appropriate to compare our method with the spectral compression of [**63**], since the latter method also preserves the original mesh connectivity, but currently, it is infeasible to apply this method to meshes with more than a few thousands of vertices, because it requires computing the eigenvectors of the mesh Laplacian matrix on both the encoder and the decoder side.

5.6.3. Solving least-squares problems. Decompressing a mesh function in the high-pass quantization method requires solving a linear least-squares problem. Sorkine et al. discussed this important algorithmic issue only briefly. To allow the reader a broader perspective on this issue, we survey here state-of-the-art least-square solvers. We briefly mention some key algorithms, provide some sample performance data, and explain how the quantization and compression methods can be tailored to ensure fast decompression. For a more complete discussion of algorithms for sparse linear least-squares problems, see Björck's monograph [**11**].

Model	Number of vertices	Relative wavelet filesize (%)	Relative highpass filesize (%)
Rabbit	107,522	0.354	0.895
Bunny	118,206	0.429	0.662
Horse	112,642	0.321	0.457
Venus	198,658	0.336	0.543
Feline	258,046	0.389	0.790

FIGURE 5.6.3. Comparison between our geometry encoding and the wavelet encoder of [**66**]. The file sizes are displayed in percents, relative to the uncompressed mesh geometry. The models were compressed by both methods with approximately the same visual error in the order of 10^{-4}, so that the compressed mesh is indistinguishable from the original.

Sparse least-squares solvers fall into two categories, direct and iterative. Most direct solvers factor the coefficient matrix $\tilde{L}$ into a product of an orthonormal matrix Q and an upper triangular matrix R, $\tilde{L} = QR$. Once the factorization is computed, the minimizer $\hat{x}$ of $\left\|\tilde{L}x - b\right\|_2$ is found by solving the triangular linear system of equations $R\hat{x} = Q^Tb$. This algorithm is backward stable. The matrix R is typically very sparse, although not as sparse as $\tilde{L}$; it is represented explicitly in such algorithms. In particular, since in our case the meshes are almost planar graphs and have small vertex separators, R is guaranteed to remain sparse [**39**]. The matrix Q is not as sparse, but it is has a sparse representation as a product of elementary orthogonal factors [**36, 38**]. To reduce the work and storage required for the factorization, the columns of the input matrix $\tilde{L}$ are usually reordered prior to the factorization [**18, 27, 37, 56**].

Another class of direct solvers, which is normally considered numerically unstable, uses a triangular factorization of the coefficient matrix $\tilde{L}^T\tilde{L}$ of the so-called normal equations. Once triangular factor R is found (it is mathematically the same R as in the $\tilde{L} = QR$ factorization), the minimizer is found by solving two triangular linear systems of equations, $R^T(R\hat{x}) = \tilde{L}^Tb$. This procedure is faster than the QR procedure, but produces less accurate solutions, because solving the normal equations is not backward stable. However, the accuracy of the solutions depends on the condition number of $\tilde{L}$ (ratio of extreme singular values), and as we have shown in Section 5.5, the matrix $\tilde{L}$ is well-conditioned thanks to the anchors, so in this case solving the normal-equations problem yields accurate solutions.

The running times and storage requirements of direct solvers can be further reduced by cutting the mesh into patches, as proposed by Karni and Gotsman [**63**], and solving on each patch separately. All the boundary vertices are then considered anchors, to ensure that the solutions on different patches are consistent. We believe that this optimization would usually be unnecessary, and that problems involving entire meshes can be solved efficiently, but we mention it as a way of handling extremely large cases. Note that to ensure that the patches are consistent, the k-anchor invertible Laplacian would need to be used here, not the k-anchor rectangular Laplacian.

In all direct methods, the factorization is computed once and used to solve for multiple mesh functions. Most of the time is spent in computing the factorization, and the cost of solving for a minimizer is negligible. Therefore, the cost of decompression using these methods is almost independent of the number of mesh functions (x, y, z, and perhaps other information, such as color).

Direct methods are fast. Table 5.6.4 records the solution times for the models used in our experiments. The table shows the time to decompose the coefficient matrix of the normal equations into its triangular factors, and the subsequent solution time for one mesh function. For example, computing the triangular factorization of the horse, a model with 19,851 vertices, took 0.9 seconds on a 2.4 GHz Pentium 4 computer, and solving for a single mesh function took 0.032 seconds once the factorization has been computed. The linear solver that we used for these experiments is TAUCS version 2.2 [**90**], which uses internally two additional libraries, ATLAS version 3.4.1 [**95**] and METIS version 4.0 [**64**]. TAUCS and METIS were compiled using the Intel C/C++ compiler version 7.1 for Linux, and ATLAS was compiled using GCC version 2.95.2. The options to the compilers included optimization options (-O3) and Pentium-4-specific instructions (-xW for the Intel compiler and inlined assembly language in ATLAS). For additional performance evaluations of TAUCS, see [**78, 61**]. We did not have a code of similar performance for computing the sparse QR factorization, but we estimate that it should be about 4–6 times slower.

Even though direct methods are fast, their running times usually scale superlinearly with the size of the mesh. Iterative least-squares solvers, which do not factor the coefficient matrix, sometimes scale better than direct methods. Perhaps the most widely-used least-squares iterative solver is LSQR, which is based on a Krylov bidiagonalization procedure [**74, 75**]. Other popular solvers include CGLS, a conjugate-gradients algorithm for solving the normal equations [**12, 32**], and CRAIG, an error-minimization bidiagonalization procedure [**26**]; see also [**75, 82**].

The convergence of these methods depends on the distribution of the singular values of the coefficient matrix $\tilde{L}$, as well as on the initial approximation. In our case $\tilde{L}$ is always well-conditioned, so we can expect

Model	Number of vertices	Factorization (sec.)	Solving (sec.)
Eight	2,718	0.085	0.004
Twirl	5,201	0.098	0.006
Horse	19,851	0.900	0.032
Fandisk	20,111	1.091	0.040
Camel	39,074	2.096	0.073
Venus	50,002	3.402	0.112
Max Planck	100,086	7.713	0.240

FIGURE 5.6.4. Running times of solving the linear least-squares systems for the different models. Most time is spent on the factorization of the coefficient matrix, which can be done during the transmission of the δ-coordinates. Solving for a single mesh function (x, y or z) takes only a negligible amount of time (see rightmost column). The experimental setup is described in the text.

reasonably rapid convergence. Furthermore, the decoder knows the values of the mesh function at the anchor vertices. By interpolating these values at non-anchor vertices, the decoder can quickly produce a good initial approximation (note, however, that even at the anchor points, the known values of the original mesh function need not coincide with the values of the least-squares minimizer).

The iterative methods mentioned above can be accelerated by using a preconditioner, (informally, an approximate inverse of $\tilde{L}$). The relationship of our coefficient matrix $\tilde{L}$ to a graph Laplacian can probably be exploited when constructing a preconditioner, since highly effective preconditioners have been discovered for Laplacians. The most important classes of such preconditioners are algebraic multigrid preconditioners [**19**], incomplete Cholesky preconditioners [**53, 72**], and more recently, support preconditioners [**16, 21, 93**]. For further information about iterative solvers and preconditioning, see [**4, 6, 11, 81**].

5.7. Conclusions

In this chapter, we have shown that it is possible to rigorously bound the error in a lossy compression method for three-dimensional meshes. The chapter focuses on one particular compression method, that presented by [**85**], but our analysis technique is probably applicable to a range of methods using similar matrices. In particular, our analysis also sheds light on the errors in a more recent compression method [**84**], in which the encoder does not send the δ-coordinates to the decoder at all, only the anchor vertices. It is also useful for analyzing Laplacian-based mesh editing techniques [**71**].

Our analysis bounds the total error generated by high-pass quantization, both the quantization component of the error and the rounding component. In other words, it accounts for the fact that the decoder uses floating-point arithmetic to reconstruct the mesh. On the other hand, our analysis does not cover the shape of the errors. Empirical results show that the error is smooth, and therefore visually acceptable; these results are consistent with other applications of the small eigenvectors of Laplacians.

Our analysis yields an error bound that is easy to compute, as we have shown in Section 5.6.1. This leads to two algorithmic benefits, in addition to the insight on why high-pass quantization works. First, it can be used by an encoder to quickly encode a mesh to a prescribed error bound. That is, the encoder can quantize the coordinates and then add anchors until the computed error bound drops below a prescribed threshold. Since our error bound is not tight, the actual error will usually be smaller than that prescribed. The encoding might not be as economical as possible, but it will be produced quickly and it will satisfy the prescribed error bound. Second, the computed bound can drive the anchor selection algorithm, as shown in Section 5.6.1.

We have used three algebraic techniques to prove our error bound. Two of them are quite novel. Our bound on the small eigenvalue of an invertible Laplacian is a relatively standard application of an area of combinatorial matrix theory called support theory, but the technique of separating of W into W_1 and W_2 is new. The main algebraic novelty in the chapter lies in the application of support theory to the analysis of the spectrum of rectangular matrices. The analysis of the rounding error is relatively straightforward.

Our research raises a number of interesting open problems for future research.

(1) Can one rigorously analyze the behavior of the eigenvectors of the Laplacian of 3D meshes? Our method works because for a vector x of mesh coordinates, $\|Lx\|$ tends to be much smaller than $\|x\|$. This happens because most of the energy of x is concentrated in the subspace of R^n that is spanned by the eigenvectors of L that correspond to small eigenvalues. But does this always happen? The answer depends on the relationship between the eigenvectors of the Laplacian and typical mesh-coordinate vectors. Ben-Chen and Gotsman [**7**] have done the first step towards resolving this question. They have shown that under certain probabilistic assumptions on the shape of 3D meshes, most of the energy of the mesh-coordinate vectors indeed lies in the subspaces spanned by the small eigenvectors. Another analysis, done by Guattery and

Miller [**50**] in a different context, may provide another perspective on the issue.

(2) Our bound on the small singular values of k-anchor Laplacian uses a maximal congestion-dilation-contention metric on an embedding of paths from all the vertices to the anchors. It is probably possible to derive other computable bounds that might sometimes be tighter, such as a bound that depends on average dilation of this embedding.

(3) Can one solve the least-squares problems that arise in our method in time linear or almost linear in the size of the mesh? We have demonstrated reasonably small running times even for large meshes, but our solution method scales superlinearly. It would be useful to find solution methods with better scaling. Algebraic multigrid methods can almost certainly solve the invertible k-anchor Laplacian equations in $O(n)$ work. We are not yet sure whether algebraic multigrid methods can also effectively solve the least-squares problem arising from the rectangular Laplacian. Another direction might be an iterative solver, such as LSQR or CGLS, coupled with an effective preconditioner. In particular, it would be interesting to know whether graph-theoretical preconditioners, such as support-tree [**41, 42**] and support-graph [**9, 13, 88, 93**] preconditioners can be adapted to this problem.

(4) Could weighted Laplacian matrices work better than unweighted Laplacians? What is the best tradeoff between the quality of the compression and the number of bits needed to represent the weights of the weighted Laplacian?

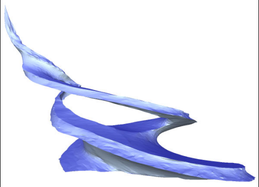
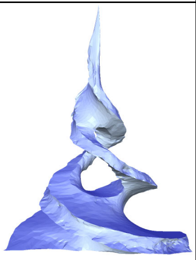
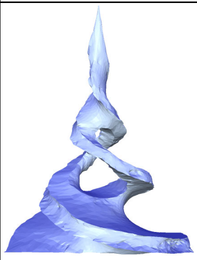
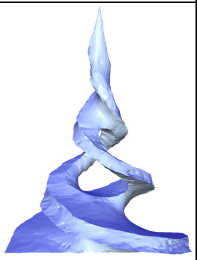
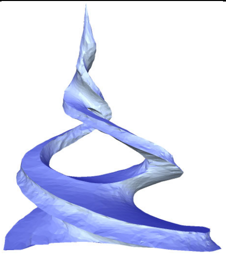
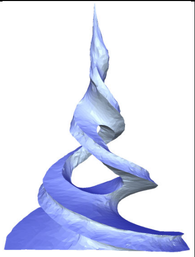
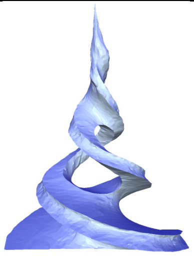
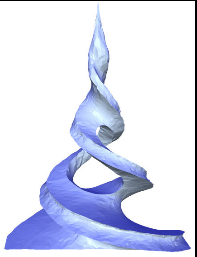
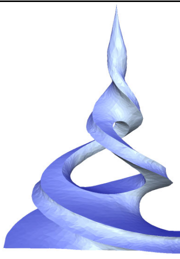
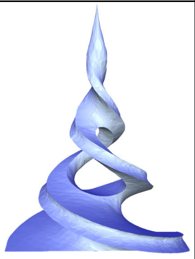
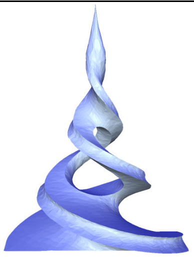
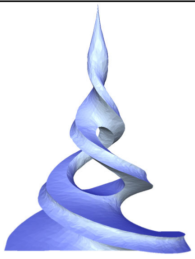
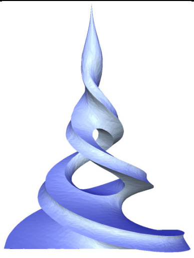
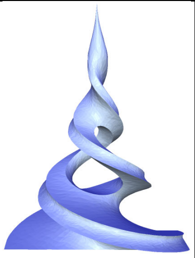
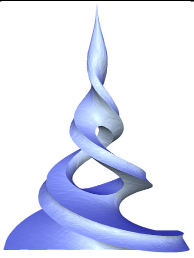
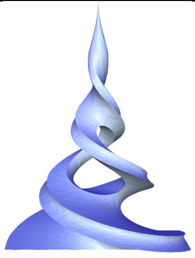
	1 anchor	15 anchors	30 anchors	45 anchors
σ_{min} bound	$5.1 \cdot 10^{-6}$	$2.0 \cdot 10^{-4}$	$4.1 \cdot 10^{-4}$	$7.8 \cdot 10^{-4}$
6 bits	 $M_q = 34.59$ $S_q = 0.17$ 5.87KB	 $M_q = 1.82$ $S_q = 0.15$ 5.95KB	 $M_q = 1.21$ $S_q = 0.14$ 6.04KB	 $M_q = 0.92$ $S_q = 0.14$ 6.13KB
7 bits	 $M_q = 21.25$ $S_q = 0.11$ 7.71KB	 $M_q = 1.06$ $S_q = 0.09$ 7.80KB	 $M_q = 0.71$ $S_q = 0.09$ 7.89KB	 $M_q = 0.48$ $S_q = 0.09$ 7.98KB
8 bits	 $M_q = 7.16$ $S_q = 0.05$ 9.45KB	 $M_q = 0.49$ $S_q = 0.05$ 9.53KB	 $M_q = 0.28$ $S_q = 0.05$ 9.62KB	 $M_q = 0.21$ $S_q = 0.05$ 9.71KB
9 bits	 $M_q = 2.55$ $S_q = 0.02$ 11.43KB	 $M_q = 0.26$ $S_q = 0.02$ 11.51KB	 $M_q = 0.15$ $S_q = 0.02$ 11.60KB	 $M_q = 0.10$ $S_q = 0.02$ 11.69KB

FIGURE 5.7.1. Visual table of quantization results for the *Twirl* model (5201 vertices). See text for details.

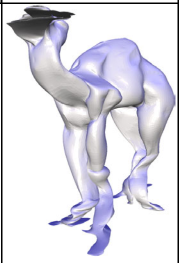
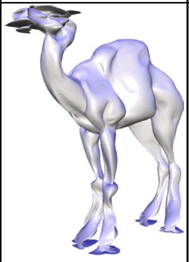
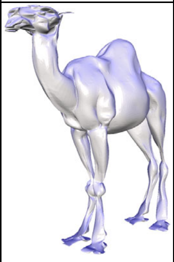
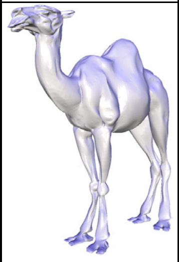
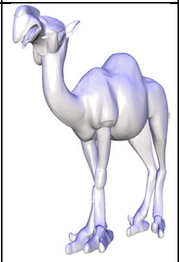
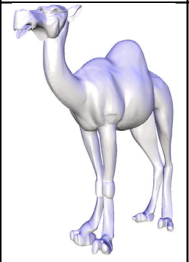
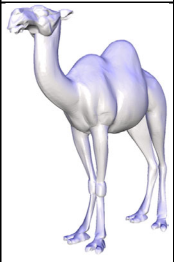
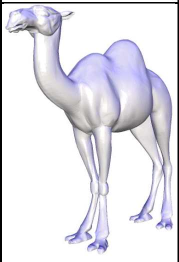
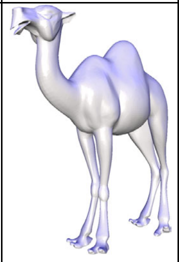
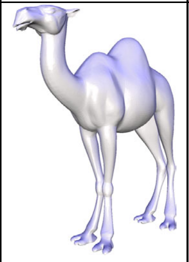
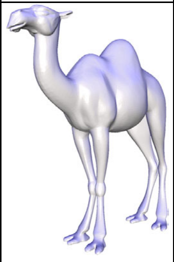
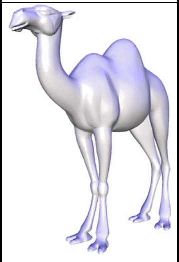
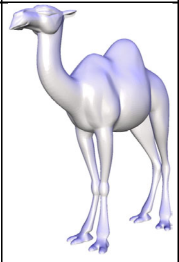
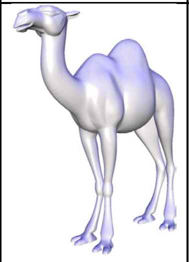
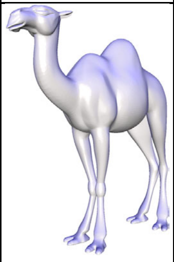
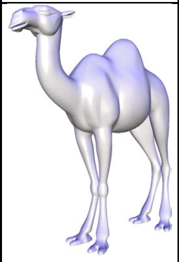
	20 anchors	60 anchors	200 anchors	400 anchors
σ_{min} bound	$7.8 \cdot 10^{-6}$	$5.7 \cdot 10^{-5}$	$3.7 \cdot 10^{-4}$	$8.3 \cdot 10^{-4}$
2 bits	 $M_q = 15.44$ $S_q = 0.10$ 13.35KB	 $M_q = 5.05$ $S_q = 0.10$ 13.60KB	 $M_q = 2.26$ $S_q = 0.09$ 14.49KB	 $M_q = 4.27$ $S_q = 0.09$ 15.76KB
3 bits	 $M_q = 6.24$ $S_q = 0.06$ 23.97KB	 $M_q = 2.54$ $S_q = 0.06$ 24.23KB	 $M_q = 0.95$ $S_q = 0.05$ 25.12KB	 $M_q = 0.63$ $S_q = 0.05$ 26.39KB
4 bits	 $M_q = 2.60$ $S_q = 0.02$ 36.79KB	 $M_q = 0.60$ $S_q = 0.02$ 37.04KB	 $M_q = 0.28$ $S_q = 0.02$ 37.93KB	 $M_q = 0.19$ $S_q = 0.02$ 39.20KB
5 bits	 $M_q = 0.62$ $S_q = 0.01$ 51.09KB	 $M_q = 0.22$ $S_q = 0.01$ 51.34KB	 $M_q = 0.10$ $S_q = 0.01$ 52.23KB	 $M_q = 0.07$ $S_q = 0.01$ 53.50KB

FIGURE 5.7.2. Visual table of quantization results for the *Camel* model (39074 vertices). See text for details.

Bibliography

[1] Pierre Alliez and Mathieu Desbrun. Valence-driven connectivity encoding for 3D meshes. *Computer Graphics Forum*, 20(3):480–489, 2001.

[2] Pierre Alliez and Craig Gotsman. Recent advances in compression of 3D meshes. In N.A. Dodgson, M.S. Floater, and M.A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, pages 3–26. Springer-Verlag, 2005.

[3] Peter Arbenz and Zlatko Drmač. On positive semidefinite matrices with known null space. *SIAM J. Matrix Anal. Appl.*, 24(1):132–149, 2002.

[4] Owe Axelsson. *Iterative Solution Methods.* Cambridge University Press, 1994.

[5] G.P. Barker and D. Carlson. Cones of diagonally dominant matrices. *Pacific J. Math.*, 57(1):15–32, 1975.

[6] R. Barret, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* Philadeplphia, PA, 1993.

[7] Mirela Ben-Chen and Craig Gotsman. On the optimality of spectral compression of mesh data. *ACM Trans. Graph.*, 24(1):60–80, 2005.

[8] A. Berman and R.J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences.* SIAM, Philadelphia, 1994.

[9] Marshall Bern, John R. Gilbert, Bruce Hendrickson, Nhat Nguyen, and Sivan Toledo. Support-graph preconditioners. Submitted to SIAM Journal on Matrix Analysis and Applications, 29 pages, January 2001.

[10] Dennis S. Bernstein. *Matrix Mathematics: theory, facts and formulas with application to linear systems theory.* Princeton University Press, 2005.

[11] Å. Björck. *Numerical Methods for Least Squares Problems.* SIAM, Philadelphia, 1996.

[12] Å. Björck and T. Elfving. Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations. *BIT*, 19:145–163, 1979.

[13] E. G. Boman, D. Chen, B. Hendrickson, and S. Toledo. Maximum weight basis preconditioners. *Numerical Linear Algebra with Applications*, (11):695–721, 2004.

[14] Erik Boman and Bruce Hendrickson. On spanning tree preconditioners. Manuscript, Sandia National Laboratories, 2001.

[15] Erik G. Boman, Doron Chen, Ojas Parekh, and Sivan Toledo. On the factor-width and symmetric H-matrices. *Linear Algebra and its Applications*, (405):239–248, 2005.

[16] Erik G. Boman and Bruce Hendrickson. Support theory for preconditioning. *SIAM Journal on Matrix Analysis and Applications*, 25(3):694–717, 2003.

[17] R. Boppana. Eigenvalues and graph bisection: An average-case analysis. In *28th Annual Symposium on Foundations of Computer Science*, pages 280–285, Los Angeles, October 1987. IEEE.

[18] Igor Brainman and Sivan Toledo. Nested-dissection orderings for sparse LU with partial pivoting. *SIAM Journal on Matrix Analysis and Applications*, 23:998–112, 2002.

[19] A. Brandt, S. F. McCormick, and J. Ruge. Algebraic multigrid (AMG) for sparse matrix equations. In D. J. Evans, editor, *Sparsity and its Applications*, pages 257–284. Cambridge University Press, 1984.

[20] William L. Briggs, Van Emden Henson, and Steve F. McCormick. *A Multigrid Tutorial*. SIAM, Philadelphia, 2 edition, 2000.

[21] Doron Chen and Sivan Toledo. Vaidya's preconditioners: Implementation and experimental study. *Electronic Transactions on Numerical Analysis*, 16:30–49, 2003.

[22] Doron Chen and Sivan Toledo. Combinatorial charaterization of the null spaces of symmetric H-matrices. *Linear Algebra and its Applications*, (392):71–90, 2004.

[23] Peter H. Chou and Teresa H. Meng. Vertex data compression through vector quantization. *IEEE Transactions on Visualization and Computer Graphics*, 8(4):373–382, 2002.

[24] Fan R. K. Chung. *Spectral Graph Theory*. American Methematical Society, 1997.

[25] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 1990.

[26] E. J. Craig. The n-step iteration procedure. *J. Math. Phys.*, 34:65–73, 1955.

[27] Timothy A. Davis, John R. Gilbert, Stefan I. Larimore, and Esmond G. Ng. A column approximate minimum degree ordering algorithm. *ACM Trans. Math. Softw.*, 30(3):353–376, 2004.

[28] James Demmel and Plamen Koev. Accurate SVDs of polynomial vandermonde matrices involving orthonormal polynomials. To appear in Linear Algebra Appl.

[29] James Demmel and Plamen Koev. The accurate and efficient solution of a totally positive generalized vandermonde linear system. *SIAM J. Matrix Anal. Appl.*, 27(1):142–152, 2005.

[30] Persi Diaconis and Daniel Stroock. Geometric bounds for eigenvalues of markov chains. *Annals of Applied Probability*, 1:36–61, 1991.

[31] W. E. Donath and A. J. Hoffman. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, 17:420–425, 1973.

[32] T. Elfving. On the conjugate gradient method for solving linear least squares problems. Tech. Report LiTH-MAT-R-78-3, Linköping University, Sweden, 1978.

[33] J. Falkner, F. Rendl, and H. Wolkowicz. A computational study of graph partitioning. *Mathematical Programming*, 66:211–239, 1994.

[34] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23:298–305, 1973.

[35] M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25:619–633, 1975.

[36] J. A. George and M. T. Heath. Solution of sparse linear least squares problems using Givens rotations. *Linear Algebra Appl.*, 34:69–83, 1980.

[37] J. A. George and E. G. Ng. On row and column orderings for sparse least squares problems. *SIAM J. Numer. Anal.*, 20:326–344, 1983.

[38] J. A. George and E. G. Ng. Orthogonal reduction of sparse matrices to upper triangular form using Householder transformations. *SIAM J. Sci. Statist. Comput.*, 7:460–472, 1986.

[39] J. A. George and E. G. Ng. On the complexity of sparse QR and LU factorization of finite-element matrices. *SIAM J. Sci. Statist. Comput.*, 9:849–861, 1988.

[40] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.

[41] K.D. Gremban, G.L. Miller, and M. Zagha. Performance evaluation of a parallel preconditioner. In *9th International Parallel Processing Symposium*, pages 65–69, Santa Barbara, April 1995. IEEE.

[42] Keith D. Gremban. *Combinatorial Preconditioners for Sparse, Symmetric, Diagonally Dominant Linear Systems*. PhD thesis, School of Computer Science, Carnegie Mellon University, October 1996. Technical Report CMU-CS-96-123.

[43] Jonathan L. Gross. Voltage graphs. *Discrete Mathematics*, 9:239–246, 1974.

[44] Jonathan L. Gross and Thomas W. Tucker. Generating all graph coverings by permutation voltage assignments. *Discrete Mathematics*, 18:273–283, 1977.

[45] Jerrold W. Grossman, Devadatta M. Kulkarni, and Irwin E. Schochetman. Algebraic graph theory without orientation. *Linear Algebra and its Applications*, 212/213:289–307, 1994.

[46] Jerrold W. Grossman, Devadatta M. Kulkarni, and Irwin E. Schochetman. On the minors of an incidence matrix and its Smith Normal Form. *Linear Algebra and its Applications*, 218(1–3):213–224, 1995.

[47] S. Guattery and G. Miller. On the performance of spectral graph partitioning methods. pages 233–242, January 1995.

[48] Stephen Guattery. On the quality of spectral separators. *SIAM J. Matrix Anal. Appl.*, 19(3):701–719, 1998.

[49] Stephen Guattery, Tom Leighton, and Gary L. Miller. The path resistance method for bounding the smallest nontrivial eigenvalue of a Laplacian. *Combinatorics, Probability, and Computing*, 8:441–460, 1999.

[50] Stephen Guattery and Gary L. Miller. Graph embeddings and Laplacian eigenvalues. *SIAM Journal on Matrix Analysis and Applications*, 21:703–723, 2000.

[51] Stefan Gumhold. New bounds on the encoding of planar triangulations. Technical Report WSI–2000–1, Wilhelm-Schickard-Institut für Informatik, University of Tübingen, Germany, January 2000.

[52] Igor Guskov, Kiril Vidimče, Wim Sweldens, and Peter Schröder. Normal meshes. In *Proceedings of ACM SIGGRAPH 2000*, pages 95–102, 2000.

[53] I. Gustafsson. A class of first-order factorization methods. *BIT*, 18:142–156, 1978.

[54] L. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. volume 11, pages 1074–1085, 1992.

[55] F. Harary. On the notion of balance of a signed graph. *Michigan Math.*, 2:143–146, 1953–1954.

[56] P. Heggernes and P. Matstoms. Finding good column orderings for sparse QR factorizations. Tech. Report LiTH-MAT-1996-20, Department of Mathematics, Linköping University, Sweden, 1996.

[57] M. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *National Bureau of Standards Journal of Research*, 49:409–436, 1952.

[58] Nicholas J. Higham. The accuracy of floating point summation. *SIAM J. Sci. Comput.*, 14(4):783–799, 1993.

[59] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, 2 edition, 2002.

[60] R.D. Hill and S.R. Waters. On the cone of positive semidefinite matrices. *Lin. Alg. Appl.*, 90:81–88, 1987.

[61] Dror Irony, Gil Shklarski, and Sivan Toledo. Parallel and fully recursive multifrontal supernodal sparse Cholesky. *Future Generation Computer Systems*, 20(3):425–440, 2004.

[62] Nabil Kahale. A semidefinite bound for mixing rates of Markov chains. *Random Structures and Algorithms*, 11:299–313, 1997.

[63] Zachi Karni and Craig Gotsman. Spectral compression of mesh geometry. In *Proceedings of ACM SIGGRAPH 2000*, pages 279–286, July 2000.

[64] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20:359–392, 1998.

[65] Andrei Khodakovsky, Pierre Alliez, Mathieu Desbrun, and Peter Schröder. Near-optimal connectivity encoding of 2-manifold polygon meshes. *Graph. Models*, 64(3/4):147–168, 2002.

[66] Andrei Khodakovsky, Peter Schröder, and Wim Sweldens. Progressive geometry compression. In *Proceedings of ACM SIGGRAPH 2000*, pages 271–278, 2000.

[67] Plamen Koev. Accurate eigenvalues and SVDs of totally nonnegative matrices. *SIAM J. Matrix Anal. Appl.*, 27(1):1–23, 2005.

[68] Toshiyuki Kohno, Hiroshi Niki, Hideo Sawami, and Yi-Ming Gao. An iterative test for H-matrix. *Journal of Computational and Applied Mathematics*, 115:349–355, 2000.

[69] Bishan Li, Lei Li, Masunore Harada, Hiroshi Niki, and Michal J. Tsatsomeros. An iterative criterion for H-matrices. *Linear Algebra and its Applications*, 271:179–190, 1998.

[70] Lei Li. On the iterative criterion for generalized diagonally dominant matrices. *SIAM J. Matrix Anal. Appl.*, 24:17–24, 2002.

[71] Yaron Lipman, Olga Sorkine, Daniel Cohen-Or, David Levin, Christian Rössl, and Hans-Peter Seidel. Differential coordinates for interactive mesh editing. In *Proceedings of Shape Modeling International*, pages 181–190, 2004.

[72] J. A. Meijerink and H. A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Mathemathics of Computation*, 31:148–162, 1977.

[73] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12:617–629, 1975.

[74] C. C. Paige and M. A. Saunders. Algorithm 583 LSQR: Sparse linear equations and sparse least squares. *ACM Trans. Math. Software*, 8:195–209, 1982.

[75] C. C. Paige and M. A. Saunders. LSQR. An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Software*, 8:43–71, 1982.

[76] A. Pothen, H. D. Simon, and K. P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11:430–452, 1990.

[77] F. Rendl and H. Wolkowicz. A projection technique for partitioning the nodes of a graph. *Annals of Operations Research*, 58:155–180, 1995.

[78] Vladimir Rotkin and Sivan Toledo. The design and implementation of a new out-of-core sparse Cholesky factorization method. *ACM Trans. Math. Software*, 30(1):19–46, 2004.

[79] Konstantin Rybnikov and Thomas Zaslavsky. Criteria for balance in abelian gain graphs with applications to geometry. 2002.

[80] Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.

[81] Yousef Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing Company, 1996.

[82] M. A. Saunders. Solution of sparse rectangular systems using LSQR and CRAIG. *BIT*, 35:588–604, 1995.

[83] Alistair Sinclair. Improved bounds for mixing rates of Markov chains and multicommodity flow. *Combinatorics, Probability, and Computing*, 1:351–370, 1992.

[84] Olga Sorkine, Daniel Cohen-Or, Dror Irony, and Sivan Toledo. Geometry-aware bases for shape approximation. *IEEE Transactions on Visualization and Computer Graphics*, 11(2):171–180, 2005.

[85] Olga Sorkine, Daniel Cohen-Or, and Sivan Toledo. High-pass quantization for mesh encoding. In *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 42–51, 2003.

[86] Daniel A. Spielman. Private communication. March 2005.

[87] Daniel A. Spielman and Shang-Hua Teng. Spectral partitioning works: Planar graphs and finite element meshes. In *IEEE Symposium on Foundations of Computer Science*, pages 96–105, 1996.

[88] Daniel A. Spielman and Shang-Hua Teng. Solving sparse, symmetric, diagonally-dominant linear systems in time $o(m^{1.31})$. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 416–427, Cambridge, MA, October 2003.

[89] Daniel A. Spielman and Shang-Hua Teng. Nearly linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. July 2004. Preprint.

[90] Sivan Toledo. TAUCS*: A Library of Sparse Linear Solvers, version 2.2.* Tel-Aviv University, Available online at http://www.tau.ac.il/~stoledo/taucs/, September 2003.

[91] Costa Touma and Craig Gotsman. Triangle mesh compression. In *Graphics Interface '98*, pages 26–34, June 1998.

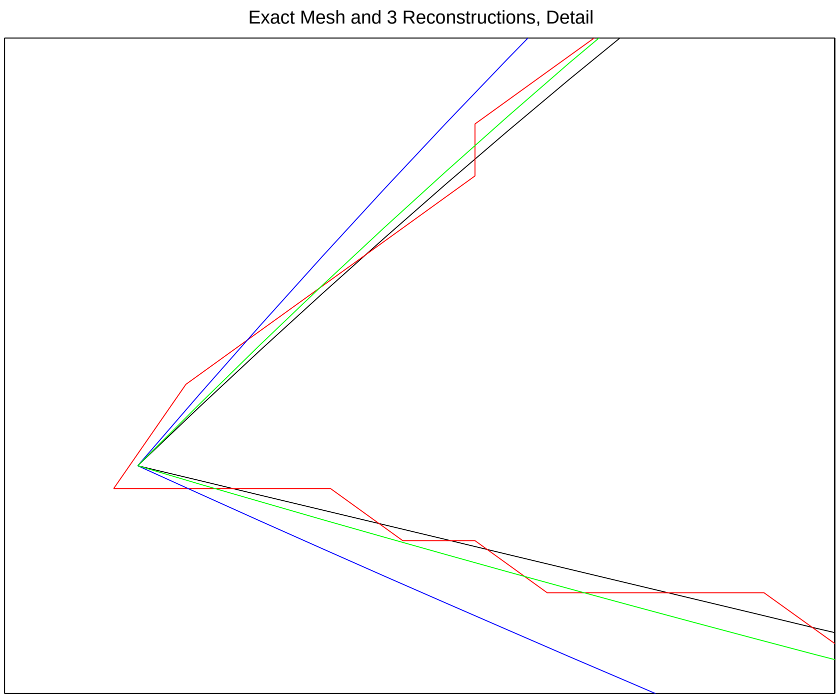
[92] Lloyd N. Trefethen and David Bau, III. *Numerical Linear Algebra.* SIAM, Philadelphia, 2000.

[93] Pravin M. Vaidya. Solving linear equations with symmetric diagonally dominant matrices by constructing good preconditioners. Unpublished manuscript. A talk based on the manuscript was presented at the IMA Workshop on Graph Theory and Sparse Matrix Computation, October 1991, Minneapolis.

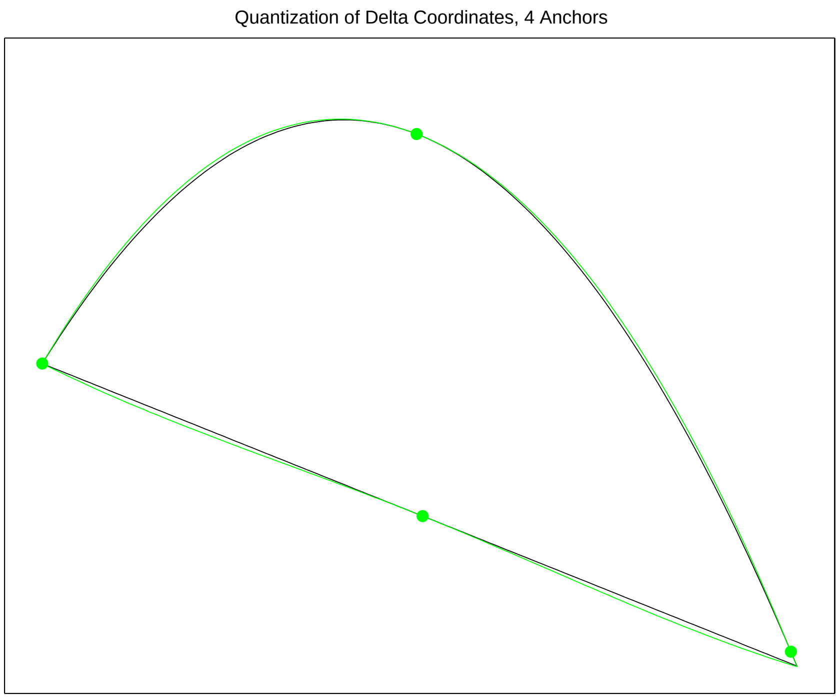
[94] R. S. Varga. On recurring theorems on diagonal dominance. *Lin. Alg. Appl.*, 13:1–9, 1976.

[95] R. Clint Whaley, Antoine Petitet, and Jack J. Dongarra. Automated empirical optimization of software and the ATLAS project. Technical report, Computer Science Department, University Of Tennessee, 2000.

[96] Thomas Zaslavsky. Signed graphs. *Discrete Appl. Math.*, 4:47–74, 1982.



איור 7: הפוליגון המקורי (בשחור) ושלושה השיחזורים מקואורדינטות ה־δ שעברו quantization. האיור מראה רק את החלק השמאלי ביותר של ארבעת הפוליגונים, כדי להראות את ההבדל האיכותי בשגיאות.



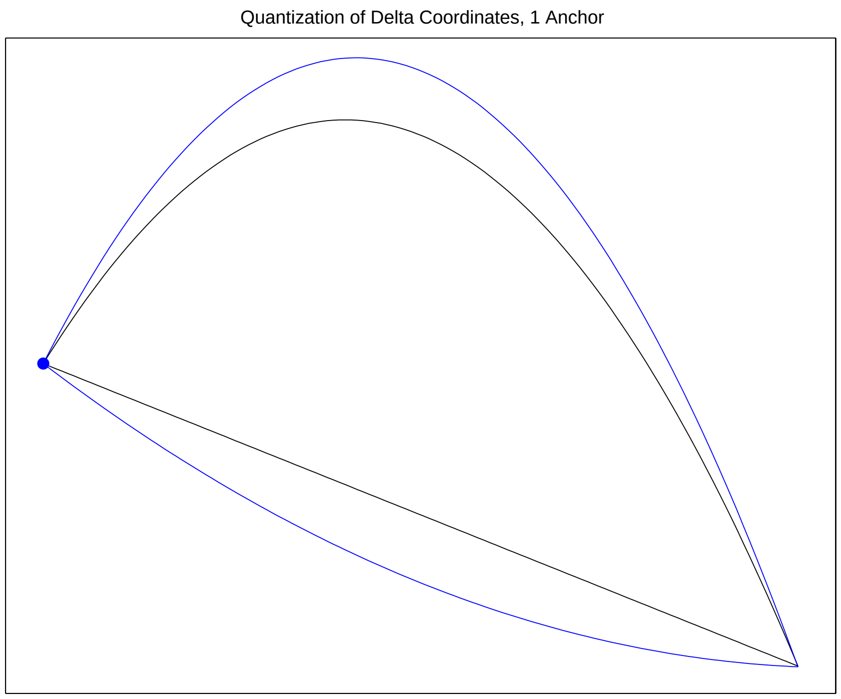
איור 6: פוליגון (בשחור) שמייצג את השריג, ושיחזור שלו (בירוק) מ־quantization של קואורדינטות ה־δ. הפעם, השיחזור עושה שימוש ב־4 עוגנים, המסומנים בעיגול.

ה־δ, לא נוכל להשתמש במערכת משוואות זאת שכן באופן כללי היא אינה קונסיסטנטית. במקום זאת, high-pass mesh quantization משחזרת קואורדינטות מקורבות באמצעות פיתרון של מערכת המשוואות במובן של ריבועים פחותים (least squares).

בפרק 5 אנו מראים שאת השגיאות הנובעות מקוואנטיזציה ואת שגיאות העיגול ניתן לחסום על־ידי פונקציה של הערך הסינגולרי הקטן ביותר של מטריצת המקדמים

$$\tilde{L} = \begin{bmatrix} 2 & -1 & & & & -1 \\ -1 & 2 & -1 & & & \\ & -1 & 2 & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & 2 & -1 \\ -1 & & & & -1 & 2 \\ 1 & & & & & \\ & & 1 & & & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ & & & & 1 & \end{bmatrix}$$

של בעיית הריבועים הפחותים. בנוסף, אנו מראים איך לחסום ערך סינגולרי זה באמצעות טיעוני תמיכה קומבינטוריים. איור 6 מראה שיחזור של הפוליגון המקורי תוך שימוש בארבעה עוגנים. איור 7 מראה תקריב של הפוליגון המקורי ושל שלושת הקירובים.



איור 5: פוליגון (בשחור) שמייצג את השריג, ושיחזור שלו (בכחול) מ-quantization של קואורדינטות ה-δ, Lx ו - Ly, שוב תוך שימוש ב-64 ערכים. השיחזור עושה שימוש בעוגן בודד, המסומן בעיגול.

קואורדינטות δ נוטות להיות קטנות יותר מקואורדינטות קרטזיות. בצורתה המטריציונית, הטרנספורמציה היא

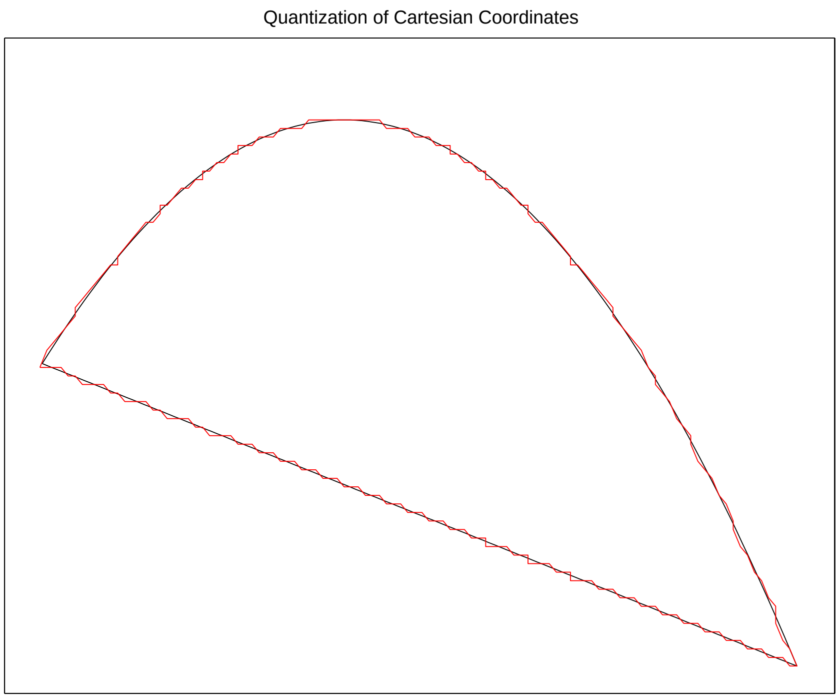
$$\delta^{(x)} = \begin{bmatrix} 2 & -1 & & & & -1 \\ -1 & 2 & -1 & & & \\ & -1 & 2 & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & 2 & -1 \\ -1 & & & & -1 & 2 \end{bmatrix} x = Lx \, .$$

מטריצת הטרנסורמציה L היא סינגולרית, כיון שסכום כל השורות הוא בדיוק 0. ניתן להפוך טרנספורמציה זאת על-ידי הוספת אילוץ אחד נוסף, למשל הקואורדינטה הקרטזית המקורית של של אחד מקודקודי השריג, למשל x_1. אנו קוראים לנקודה כזאת עוגן.

כאשר יש עוגן בודד, הטרנספורמציה הפיכה, אבל בעלת התנייה גרועה (ill-conditioned): הערך הסינגולרי הקטן ביותר של הטרנספורמציה הוא קטן, מה שמרמז שהנורמה של הטרנספורמציה ההופכית הוא גדול. זה גורם לשגיאה גדולה בקירוב, כפי שניתן לראות באיור 5. כדי לשפר את ההתנייה של הטרנספורמציה, אנו מוסיפים עוד עוגנים. קואורדינטות ה-δ יחד עם הקואורדינטות הקרטזיות של העוגנים מגדירות את הקואורדינטות המקוריות באמצעות המשוואות

$$\begin{aligned} Lx &= \delta^{(x)} \\ x_{k_1} &= b_{k_1} \\ &\vdots \\ x_{k_\ell} &= b_{k_\ell} \, , \end{aligned}$$

בהן $k_1, \ldots, k_\ell$ הם האינדקסים של העוגנים. אולם, אם ננסה לשחזר קירוב ל - x בהינתן קואורדינטות



איור 4: פוליגון (בשחור) שמייצג את השריג, ופוליגון (באדום) שנובע מ־quantization של קואורדינטות ה - x וה - y של הפוליגון תוך שימוש ב - 64 ערכים.

את הקוואנטיזציה ניתן לבצע ישירות על הקואורדינטות הקרטזיות, או על איזושהי טרנספורמציה הפיכה שלהן. ב - high-pass mesh quantization, מבצעים את הקוואנטיזציה לאחר שהקואורדינטות עברו טרנספורמציה השייכת למחלקה מסויימת של אופרטורים לינאריים [85].

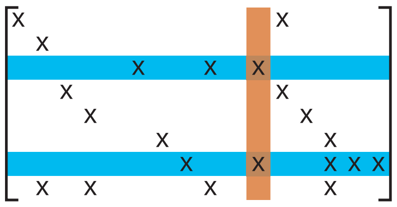
בפרק 5 אנחנו מציגים ניתוח אלגברי של High-Pass Quantization. אנו מראים כיצד ניתן לחסום אלגברית את השגיאה שנובעת משימוש בשיטת דחיסה זו. אנו מראים כי ניתן להשתמש בערך הסינגולרי הקטן של מטריצת הטרנספורמציה הן כדי לחסום את שגיאת הקירוב והן כדי לחסום את שגיאת העיגול, הנובעת משימוש באריתמטיקת floating point. בנוסף, אנו משתמשים בכלים של תורת התמיכה כדי להוכיח חסם על ערך סינגולרי זה. אנו מראים ניסויית ששיטה זאת היא אפקטיבית, ושהחסם העליון על השגיאה אינו פסימי מדי. התוצאות בפרק 5 התקבלו לפירסום ב - **ACM Transactions on Graphics**, מאת "Algebraic Analysis of High-Pass Quantization" Doron Chen, Daniel Cohen-Or, Olga Sorkine, and Sivan Toledo.

אנו משתמשים עכשיו בגירסה דו־מימדית של high-pass mesh quantization כדי להדגים שיטה זאת ואת טכניקת האנליזה שלנו. יהא m פוליגון סגור במישור. אם נתייחס אליו כאל שריג, הקשתות שלו הן פאות של השריג. כיון שהפוליגון סגור, הגרף של השריג הוא מעגל פשוט. לכל קודקוד בשריג מתאימים ערכי x ו - y. הפעלת קוואנטיזציה ישירות על ערכי x ו - y גורם לשגיאות בתדר גבוה, שאינן נעימות לעין, כפי שניתן לראות באיור 4.

על מנת להימנע מבעייה זאת, high-pass quantization מבצעת תחילה טרנספורמציה של הקואורדינטות באמצעות מטריצה דמויית־לפלסיאן. הקואורדינטות שעברו טרנספורמציה נקראות קואורדינטות־δ. הטרנספורמציה פועלת בצורה הבאה: קואורדינטת ה־δ של קודקוד i היא

$$\delta_i^{(x)} = 2x_i - x_{i-1} - x_{i+1} = \mathrm{degree}(i)\left(x_i - \frac{\sum_{j \text{ is a neighbor of } i} x_j}{\mathrm{degree}(i)}\right)$$

ובאופן דומה עבור $\delta_i^{(y)}$. הביטוי בסוגריים הוא המרחק בין קואורדינטת ה - x של i לבין הממוצע של שכניו. כאשר m מקרב עקומה חלקה c סמוך לקודקוד i, הממוצע של שכניו נוטה לנבא טוב את x_i, ולכן



איור 3: הדגמה של החסם $\|W\|_2^2 \leq \max_j \sum_{i:W_{i,j}\neq 0} \|W_{i,:}\|_2^2$. ה־X־ים באיור מייצגים איברים שונים מאפס ב־ W. החסם קובע כי $\|W\|_2^2$ חסומה על־ידי המקסימום, על כל העמודות j של W, של סכום ריבועי הנורמות־2 של שורות W שמכילות ערך שונה מאפס בעמודה ה־ j.

מדגים חסם זה. ברור כי חסם זה אינו גרוע יותר (ולעיתים קרובות טוב יותר) מחסם פרובניוס שהוא סכום הריבועים של נורמות ה־2 של כל השורות.

למרות שחסמים אלה הם כלליים, ושניתן להפעילם על כל מטריצה, המוטיבציה לאנליזה שהניבה אותם הגיעה מתורת התמיכה. כמו כן, ההוכחות של חסמים אלה משתמשות בכלים של תורת התמיכה.
איך חסמי נורמות קשורים לתורת התמיכה? Boman ו־ Hendrickson [16], וגם Boman, Chen, Hendrickson ו־ Toledo [13] הראו שכאשר מבצעים preconditioning למטריצת SDD על־ידי מטריצת SDD אחרת, קצב ההתכנסות של אלגוריתם איטרטיבי חסום על־ידי מכפלת הנורמות של שתי מטריצות, W ו־ Z. מטריצות אלו אינן מוגדרות באופן יחיד, אבל embedding של קשתות הגרף של A לתוך מסלולים בגרף של B מגדירה היטב את W, ו־ embedding של קשתות G_B לתוך מסלולים ב־ G_A מגדירה היטב את Z. כאשר W מוגדרת על־ידי embedding, ניתן לפרש חסמים ידועים על $\|W\|_2$ כמדדים קומבינטוריים של איכות ה־ embedding. למשל, נורמת פרובניוס, שחוסמת את נורמת־2, פרופורציונלית ל־ stretch הממוצע של קשתות ב־ embedding, עבור הגדרה מתאימה של stretch [14].
כאשר W מוגדרת על־ידי embedding, ניתן לפרש את החסמים החדשים שלנו על נורמת־2 כמדדים קומבינטוריים של ה־ embedding. למעשה, המוטיבציה לאנליזה שהובילה לחסמים אלה נבעה ממדד קומבינטורי חדש של embeddings בו השתמשו Spielman ו־ Teng כדי לחסום את קצב ההתכנסות של שיטות איטרטיביות שמשתמשות ב־ preconditioners.
התוצאות בפרק 4 התקבלו לפירסום ב־ **Electronic Transactions on Numerical Analysis**, "Obtaining Bounds on the Two Norm of a Matrix from the Splitting Lemma" מאת Doron Chen, John R. Gilbert and Sivan Toledo.

יישומים של תורת התמיכה בגרפיקה ממוחשבת

בפרק 5 אנו משתמשים בכלים של תורת התמיכה על־מנת לנתח טכניקת דחיסה של שריגים הנקראת high-pass quantization [85]. בגרפיקה ממוחשבת, עצמים מיוצגים לעיתים קרובות על־ידי שריגים תלת־מימדיים: קבוצה של משולשים החולקים קשתות וקודקודים. כל קשת, פרט לאלה על ה־ boundary, שייכת לשני משולשים. השריג מיוצג על־ידי קבוצה סופית של קודקודים, קבוצה סופית של משולשים, ומידע גיאומטרי. המידע הגיאומטרי מורכב, בדרך כלל, מהקואורדינטות הקרטזיות של כל קודקוד. באופן מופשט, ניתן לראות את השריג כגרף או היפר־גרף ובנוסף ווקטורים של הקואורדינטות הקרטזיות של קודקודי השריג, ווקטור x, ווקטור y ווקטור z.
דחיסת שריגים כרוכה בשתי בעיות שבדרך־כלל נפתרות בנפרד: קידוד הקישוריות וקידוד הגיאומטריה. בעוד שאלגוריתמים מתקדמים לדחיסת הקישוריות הם אפקטיביים מאד [1, 51, 65, 91], דחיסת הגיאומטריה נותרה אתגר. המידע הגיאומטרי הגולמי מגיע בדרך־כלל בייצוג floating-point בדיוק גבוה. אי אפשר לדחוס מידע כזה בשיטות דחיסה רגילות; לפיכך, רוב שיטות קידוד הגיאומטריה משתמשות בקוואנטיזציה (quantization), כלומר ייצוג ברמת דיוק נמוכה יותר, מה שגורם לשגיאות ולאיבוד חלק מהמידע.

כתוצאה מכך, הרכיב השמאלי הוא מאוזן ותורם ווקטור אחד למרחב הגרעין. הרכיב הימני אינו מאוזן ולכן דרגתו מלאה.

אנו מציגים איפיון קומבינטורי של ווקטור הגרעין של כל רכיב קשירות מאוזן. מסתבר שעל־מנת לבנות ווקטור גרעין של רכיב מאוזן, עלינו לבחור קודקוד אקראי ברכיב. נקרא לקודקוד זה שורש. יהא v ווקטור הגרעין המתאים לרכיב. אזי $v_j = 0$ אם ורק אם קודקוד j אינו ברכיב. לכל קודקוד j ברכיב הקשירות, v_j הוא הרווח של מסלול בין j לבין השורש. הרווח הזה מוגדר היטב, שכן ברכיב קשירות מאוזן לא משנה איזה מסלול נבחר בין קודקוד j לבין השורש; לכל המסלולים יש אותו הרווח.

בדוגמא שלנו, הבא נבחר בקודקוד 2 להיות השורש של רכיב הקשירות השמאלי. אזי רכיב זה תורם לגרעין את הווקטור הבא:

$$\begin{bmatrix} 0.25 \\ 1 \\ -0.125 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} .$$

מספר המעגלים בגרף הוא באופן טיפוסי רב מאד; למרות זאת, אנו מציגים אלגוריתם קומבינטורי יעיל כדי לקבוע את הדרגה וכדי לבנות בסיס אורתונורמלי למרחב הגרעין של מטריצה עם רוחב פקטור 2, בהינתן פירוק רוחב פקטור 2 שלה. אנו מראים כיצד לנצל אלגוריתם קומבינטורי זה כדי לפתור מערכות לינאריות סינגולריות מסוימות באריתמטיקת finite-precision. התוצאות בפרק 3 פורסמו ב - "Combinatorial Characterization of the Null Spaces of Symmetric H-Matrices", **Linear Algebra and its Applications 392: 71–90 (2004)** מאת Doron Chen and Sivan Toledo.

חסמים על נורמת־2 של מטריצה

בפרק 4 אנו מוכיחים מספר חסמים על נורמת־2 של מטריצה מלבנית ממשית. החסמים האלו מנצלים את הדלילות של W, והם לעיתים קרובות הדוקים יותר מחסמים ידועים אחרים כגון נורמת פרובניוס (Frobenius). תהא W מטריצה בגודל k על m. אנו מראים כי:

$$\begin{aligned}
\|W\|_2^2 &\le \max_j \sum_{i:W_{i,j}\neq 0} \|W_{i,:}\|_2^2 = \max_j \sum_{i:W_{i,j}\neq 0} \sum_{c=1}^{m} W_{i,c}^2 \\
\|W\|_2^2 &\le \max_i \sum_{j:W_{i,j}\neq 0} \|W_{:,j}\|_2^2 = \max_i \sum_{j:W_{i,j}\neq 0} \sum_{r=1}^{k} W_{r,j}^2 \\
\|W\|_2^2 &\le \max_j \sum_{i:W_{i,j}\neq 0} |W_{i,j}| \cdot \left(\sum_{c=1}^{m} |W_{i,c}| \right) \\
\|W\|_2^2 &\le \max_i \sum_{j:W_{i,j}\neq 0} |W_{i,j}| \cdot \left(\sum_{r=1}^{k} |W_{r,j}| \right) \\
\|W\|_2^2 &\le \left\| W W^T \right\|_1 \\
\|W\|_2^2 &\le \left\| W^T W \right\|_1
\end{aligned}$$

החסם הראשון קובע שהנורמה־2 של W בריבוע, חסומה על־ידי המקסימום, על כל העמודות j של W, של של סכום ריבועי הנורמות־2 של שורות W שמכילות ערך שונה מאפס בעמודה ה - j. איור 3

מרחבי גרעין של מטריצות־H סימטריות

בפרק 3 אנו מאפיינים את המבנה של מרחבי הגרעין של מטריצות עם רוחב פקטור 2 (שכפי שהראנו זוהי משפחת מטריצות H עם איברי אלכסון אי־שליליים, או מטריצות H^+ סימטריות). האיפיון מתבסס על תכונות קומבינטוריות של הגרפים המתאימים למטריצות עם רוחב פקטור 2. אנו מציגים אלגוריתמים קומבינטוריים המסוגלים לחשב את הדרגה ולבנות בסיס אורתונורמלי למרחב הגרעין של מטריצות עם רוחב פקטור 2, בהינתן פירוק רוחב פקטור 2 שלהן.

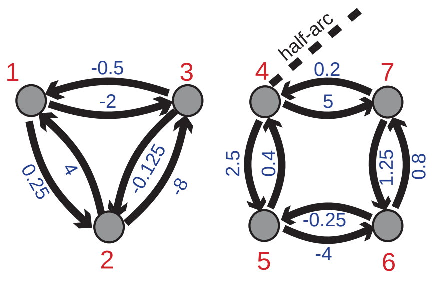
אנו חוקרים את הקשר ההדוק בין מטריצות־H סימטריות עם איברי אלכסון אי־שליליים לבין gain graphs [79, 96] (שנקראו בעבר voltage graphs [43, 44]). גרף gain הוא גרף לא מכוון שבו ניתן לראות כל קשת e כצמד קשתות, אחת בכל כיוון, עם משקלות שונים בכל כיוון. משקלות אלה נקראים רווחים (gains), ויש להן את התכונה הבאה: תהא e קשת המחברת קודקוד i עם קודקוד j. אזי ל־ e יש שני רווחים: רווח g מ־ i ל־ j, ורווח $1/g$ מ־ j ל־ i.

המחלקה של מטריצות עם רוחב פקטור 2 כוללת את כל המטריצות שניתן להציגן כ־ $A = UU^T$, כך שבכל עמודה של U יש לכל היותר שני איברים השונים מאפס. הגרף המתאים ל־ A נבנה בצורה הבאה: כל עמודה u במטריצה U מיוצגת על־ידי קשת. אם ב־ u יש שני איברים שונים מאפס, α במיקום i ו־ β במיקום j, אז נציג אותו כקשת בין קודקוד i לקודקוד j; הרווח של קשת זו יהיה $-\beta/\alpha$ בין i ל־ j, ו־ $-\alpha/\beta$ בין j ל־ i. לעמודות עם איבר אחד שונה מאפס נתאים חצי־קשת, שזו קשת אם נקודת קצה אחת בלבד וללא רווח; אם ל־ u יש ערך אחד שונה מאפס במיקום ה־ i, אז נתאים לו חצי־קשת שחלה בקודקוד ה־ i.

לדוגמא, נניח $A = UU^T$ כאשר

$$U = \begin{bmatrix} 4 & & 1 & & & & & \\ -1 & 1 & & & & & & \\ & 8 & 2 & & & & & \\ & & & 2 & & & -1 & -3 \\ & & & -5 & 2 & & & \\ & & & & 8 & 5 & & \\ & & & & & -4 & 5 & \end{bmatrix} .$$

הגרף המתאים למטריצה זאת הוא



שימו לב כי ניתן לראות את מבנה ה־ nonzeros של U כמטריצת החילה של הגרף המתאים ל־ A.

בפרק 3 אנו מראים שהמבנה של מרחב הגרעין של מטריצת־H סימטרית תלוי במבנה של רכיבי הקשירות של הגרף המתאים לה. כל רכיב קשירות תורם לכל היותר ווקטור אחד למרחב הגרעין.

בהינתן רכיב קשירות מסויים, האם הוא תורם ווקטור למרחב הגרעין או לא? ובכן, זה תלוי באיזון שלו. על־מנת להגדיר איזון של רכיב קשירות, אנו זקוקים להגדרות הבאות: הרווח של מסלול מכוון או מעגל הוא מכפלת הרווחים של הקשתות לאורך המסלול או המעגל. מעגל הוא מאוזן אם הרווח שלו הוא בדיוק $+1$. רכיב קשירות נקרא מאוזן אם הוא אינו מכיל חצאי־קשתות וגם כל המעגלים בו הם מאוזנים. הראנו כי כל רכיב מאוזן תורם ווקטור אחד למרחב הגרעין, בעוד שלכל רכיב לא מאוזן יש דרגה מלאה. בדוגמא שלנו, רכיב הקשירות השמאלי מכיל מעגל אחד שהוא מאוזן ואינו מכיל חצאי־קשתות.

$$A = \begin{bmatrix} 5 & -3 \\ -3 & 2 \end{bmatrix} \qquad |A| = \begin{bmatrix} 5 & 3 \\ 3 & 2 \end{bmatrix}$$

$$dn\left(|A|\right) = \begin{bmatrix} 1 & \frac{3}{\sqrt{10}} \\ \frac{3}{\sqrt{10}} & 1 \end{bmatrix} \qquad \lceil \|dn\left(|A|\right)\|_2 \rceil = \left\lceil \left\| \begin{bmatrix} 1 & \frac{3}{\sqrt{10}} \\ \frac{3}{\sqrt{10}} & 1 \end{bmatrix} \right\| \right\rceil = \lceil 1.9487 \rceil = 2$$

איור 2: דוגמא לחסם תחתון על רוחב הפקטור.

אינה שולטת אלכסונית.

כמו כן, ברור שלא לכל המטריצות הסימטריות החיוביות למחצה יש רוחב פקטור 2. לדוגמא,

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} .$$

את מטריצה A ניתן להציג כך:

$$A = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} ,$$

לכן רוחב הפקטור שלה הוא לכל היותר 3. כמו כן, ל - A לא יכול להיות ייצוג עם רוחב פקטור 2: הדרגה של A היא 1, לכן כל ייצוג של A כ - $A = UU^T$ חייב לקיים שכל העמודות של U זהות עד כדי מכפלה בקבוע. לכן, בלי הגבלת הכלליות, ב - U יש עמודה בודדת, וקל להראות שעמודה זו היא עמודה המורכבת רק מהספרה 1.

התוצאות המרכזיות של פרק 2 הן:

למטריצה יש רוחב פקטור לכל היותר שתיים אם ורק אם היא מטריצת-H^+ סימטרית. הסימון H^+ מייצג מטריצות-H עם איברי אלכסון אי־שליליים.

הגדרה: $dn(A)$ מסמן את ה - scaling הסימטרי של השורות והעמודות של A, שמביא לכך שהאיברים על האלכסון של A יהיו כולם 1 (מלבד איברי האלכסון של שורות מאופסות של A, שנשארים אפס).

משפט: לכל מטריצה A SPSD, רוחב הפקטור של A חסום מלמטה על־ידי $\lceil \|dn(A)\|_2 \rceil$.

הוכחנו גם חסם הדוק יותר (דוגמא לחסם זה מופיעה באיור 2):

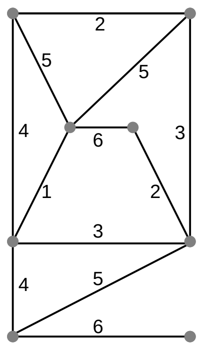
משפט: לכל מטריצה A SPSD, רוחב הפקטור של A חסום מלמטה על־ידי $\lceil \|dn\left(|A|\right)\|_2 \rceil$. הסימון $|A|$ מייצג את המטריצה שבה האיבר (i,j) הוא $|A_{ij}|$.

משפט: למטריצה A יש רוחב פקטור לכל היותר 2 אם ורק אם היא סימטרית עם איברי אלכסון אי־שליליים ומקיימת $\lceil \|dn\left(|A|\right)\|_2 \rceil \leq 2$.

היופי של התוצאות האלו נובעת מכך שהן קושרות מדד קומבינטורי בדיד, כלומר רוחב הפקטור, עם נורמה של מטריצה, שהוא מדד רציף. התוצאות בפרק 2 פורסמו ב - "On Factor Width and Symmetric H-matrices", **Linear Algebra and its Applications** 405: 239–248 (2005) מאת Erik G. Boman, Doron Chen, Ojas Parekh and Sivan Toledo.

$$A = \begin{bmatrix} 11 & -2 & -5 & & -4 & & & \\ -2 & 13 & -5 & & & -6 & & \\ -5 & -5 & 17 & -6 & -1 & & & \\ & & -6 & 8 & & -2 & & \\ -4 & & -1 & & 12 & -3 & -4 & \\ & -6 & & -2 & -3 & 16 & -5 & \\ & & & & -4 & -5 & 15 & -6 \\ & & & & & & -6 & 6 \end{bmatrix}$$

$$U = \begin{bmatrix} \sqrt{2} & \sqrt{5} & \sqrt{4} & & & & & & & & & \\ -\sqrt{2} & & & \sqrt{5} & \sqrt{6} & & & & & & & \\ & -\sqrt{5} & & -\sqrt{5} & & \sqrt{6} & \sqrt{1} & & & & & \\ & & -\sqrt{4} & & & -\sqrt{6} & & \sqrt{2} & & & & \\ & & & & & & -\sqrt{1} & & \sqrt{3} & \sqrt{4} & & \\ & & & & -\sqrt{6} & & & -\sqrt{2} & -\sqrt{3} & & \sqrt{5} & \\ & & & & & & & & & -\sqrt{4} & -\sqrt{5} & \sqrt{6} \\ & & & & & & & & & & & -\sqrt{6} \end{bmatrix}$$



איור 1: מטריצת A SDD (למעלה), הפירוק המלבני שלה U (באמצע) והגרף שלה G_A (למטה). הצמתים ממוספרים מלמעלה למטה ומשמאל לימין.

$$\begin{aligned}
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 4 \\ 0 & 4 & 4 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & -1 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & 4 \\ 0 & 4 & 4 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & -1 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 2 \\ 2 \end{bmatrix} \cdot \begin{bmatrix} 0 & 2 & 2 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & -1 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 2 \\ 2 \end{bmatrix} \cdot \begin{bmatrix} 0 & 2 & 2 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & -1 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 1 \\ -1 & 2 & 0 \\ 0 & 2 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 \\ -1 & 2 & 0 \\ 0 & 2 & 0 \end{bmatrix}^T .
\end{aligned}$$

שימו לב כי בכל צעד של הפירוק אנו מטפלים בערך אחד מחוץ לאלכסון או בערך אלכסון אחד של שורה שהיא שולטת אלכסונית ממש.

מבנה ה ־ nonzeros של U הוא זה של מטריצת החילה (incidence matrix) של הגרף המתאים ל ־A; מבנה ה ־ nonzeros של A עצמה היא זו של מטריצת סמיכויות (adjacency matrix) של אותו הגרף. איור 1 מראה מטריצה A, פירוק מלבני שלה U ואת הגרף המתאים לה.

ההתאמה בין מטריצות סימטריות שולטות־אלכסונית וגרפים ממושקלים נוצלה פעמים רבות בעבר, בין השאר בשביל:

- לאפיין תכונות של הגרף, כגון גודל מפריד צמתים מאוזן, באמצעות ערכים עצמיים של המטריצה המתאימה [17, 31, 33, 34, 35, 47, 48, 54, 76, 77, 87];
- לחסום את הערך העצמי הקטן של A באמצעות מבנים בגרף [30, 34, 35, 49, 50, 62, 83].
- לחסום את מספר ההתנייה (condition number) המוכלל של זוג מטריצות באמצעות embeddings של הגרפים המתאימים להם [9, 16, 42];
- לבנות preconditioners על־ידי דילול של הגרף המתאים [9, 13, 93] או על־ידי בניית גרף הקשור אליו [42];
- לתכנן אלגוריתמי גרפים קומבינטוריים לחישוב מרחב הגרעין של מטריצות סימטריות שולטות־אלכסונית [22].

אנו מגדירים את רוחב הפקטור של מטריצה A להיות המספר הטבעי הקטן ביותר k כך שיש ל ־ A פירוק סימטרי $A = UU^T$ כך שבכל עמודה של U יש לכל היותר k איברים השונים מאפס. אנו טוענים כי רוחב הפקטור הוא מדד טוב עבור ה"מורכבות הקומבינטורית" של מטריצה. מטריצה סימטרית שולטת־אלכסונית היא פשוטה מהבחינה שיש לה מטריצת החלה U שמתאימה לגרף ממושקל לא־מכוון. למטריצות עם רוחב פקטור 3 אין פירוק כזה, אבל יש להן מטריצת החלה U שמתאימה להיפר־גרף שבו יש לכל־היותר 3 קודקודים בכל היפר־קשת.

ברור כי לא כל המטריצות עם רוחב פקטור 2 שולטות אלכסונית. לדוגמא,

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$$

תקציר

תזה זאת מציגה תוצאות אלגבריות ואלגוריתמיות באלגברה ליניארית דלילה. כל התוצאות הללו עושות שימוש בתורת התמיכה (Support Theory).

פרק זה מדגים את התוצאות העיקריות של התזה. מטרת הפרק היא להסביר את מה הן התוצאות, ולא איך הוכחנו אותן. אנו משתמשים בדוגמאות קטנות רבות לצורך המחשה. כל סעיף מתאר את התוצאות העיקריות של פרק אחד בתזה.

רוחב פקטור

בפרק 2 אנו מגדירים מושג מטריציוני בשם רוחב פקטור (factor width), מדד למורכבות הקומבינטורית של מטריצות סימטריות חיוביות למחצה (מטריצות SPSD). מושג זה נותן לנו היררכיה של מחלקות מטריצות עבור מטריצות SPSD. מטריצות עם רוחב פקטור 2 הן מטריצות שניתן לייצגן כגרפים. מטריצות עם רוחב פקטור 3 הן מטריצות שניתן לייצגן כהיפר־גרפים, כך שכל היפר־קשת מחברת לכל היותר שלושה קודקודים. באופן כללי, מטריצות עם רוחב פקטור k הן מטריצות שניתן לייצגן כהיפר־גרפים, כך שכל היפר־קשת מחברת לכל היותר k קודקודים. ההתאמה בין מטריצות SPSD להיפר־גרפים היא הכללה של התאמה ידועה בין גרפים ממושקלים ומטריצות סימטריות שולטות־אלכסונית, שנעשה בה שימוש רב בעבר.

תמיד ניתן לפרק מטריצה ממשית סימטרית שולטת־אלכסונית (מטריצה SDD) A פירוק סימטרי $A = UU^T$, כך שלעמודות של U יש לכל היותר שני איברים שונים מאפס (ובפרט, כל האיברים השונים מאפס בעמודה של U זהים עד כדי סימן) [13]. לדוגמא, הביטו במטריצה

$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 5 & 4 \\ 0 & 4 & 4 \end{bmatrix} .$$

את מטריצה A ניתן להציג כ־ $A = UU^T$ כאשר

$$U = \begin{bmatrix} 1 & 0 & 1 \\ -1 & 2 & 0 \\ 0 & 2 & 0 \end{bmatrix} .$$

פירוק זה מחושב בדרך הבאה:

$$\begin{aligned} A &= \begin{bmatrix} 2 & -1 & 0 \\ -1 & 5 & 4 \\ 0 & 4 & 4 \end{bmatrix} = \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 4 \\ 0 & 4 & 4 \end{bmatrix} + \begin{bmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{aligned}$$

הקומבינטורית של חסמים אלה. תוצאות אלו התקבלו לפירסום ב - **Electronic Transactions on Numerical Analysis**, "Obtaining Bounds on the Two Norm of a Matrix from the Splitting Lemma" מאת Doron Chen, John R. Gilbert and Sivan Toledo.

4. יישומים של תורת התמיכה בגרפיקה ממוחשבת: אנו מציגים יישומים חדשים של תורת התמיכה בגרפיקה ממוחשבת. ברוב השימושים הקיימים של תורת התמיכה ניתנת לנו מטריצה, ואנו מייצגים אותה כגרף על־מנת לנתח אותה או על־מנת לבצע לה preconditioning. ביישום זה, לעומת זאת, התהליך הוא הפוך. במקרה זה ניתן לנו גרף שברצוננו לדחוס. הדחיסה עושה שימוש במטריצה דמוית לפלסיאן שנובעת מגרף הקשירות.
אנחנו מבצעים ניתוח אלגברי של שיטת דחיסת שריגים הנקראת High-Pass Quantization. ב - High-Pass Quantization, מטריצה מלבנית שמבוססת על הלפלסיאן של השריג מופעלת על הווקטור של הקואורדינאטות הקרטזיות של קודקודי השריג. את הווקטורים המתקבלים, שנקראים קואורדינאטות־δ, מייצגים בדיוק נמוך יותר. המטריצה שמופעלת נגזרת מהטופולוגיה של השריג ומהאינדקסים של קבוצה קטנה של צמתים בשריג (עוגנים), אבל לא מהקואורדינאטות של צמתים אלה. ניתן לקבל קירוב של הגיאומטריה מקואורדינטות ה־δ המקורבות ומהמיקום המרחבי של העוגנים. אנו מראים כיצד ניתן לחסום אלגברית את השגיאה שנובעת משימוש בשיטת דחיסה זו. אנו מראים כי ניתן להשתמש בערך הסינגולרי הקטן של מטריצת הטרנספורמציה הן כדי לחסום את שגיאת הקירוב והן כדי לחסום את שגיאת העיגול, הנובעת משימוש באריתמטיקת floating point. בנוסף, אנו מראים חסם על ערך סינגולרי זה. החסם הוא פונקציה של הטופולוגיה של השריג ושל העוגנים שנבחרו. אנו מציעים גם אלגוריתם לבחירת עוגנים, שנועד להקטין חסם זה. אנו מראים בצורה ניסויית כי אלגוריתם זה הוא יעיל, וכי החסם העליון שחישבנו לשגיאה אינו פסימי מדי. תוצאות אלו התקבלו לפירסום ב - **ACM Transactions on Graphics**, "Algebraic Analysis of High-Pass Quantization" מאת Doron Chen, Daniel Cohen-Or, Olga Sorkine, and Sivan Toledo.

תמצית

תזה זאת מציגה תוצאות אלגבריות ואלגוריתמיות באלגברה ליניארית דלילה. כל התוצאות הללו עושות שימוש בתורת התמיכה (Support Theory). למרות שהכלים של תורת התמיכה נועדו במקור על־מנת לחסום את מספרי ההתניה (condition numbers) של מערכות ליניאריות שעברו preconditioning, אנו מראים כי לכלים אלה יש מגוון של שימושים אחרים. לתוצאות בעבודה זאת יש יישומים בתורת התמיכה, אך ניתן לראות אותן גם כתוצאות אלגבריות עצמאיות. למשל, אנו מראים כיצד ניתן להשתמש בתורת התמיכה כדי להשיג חסמים חדשים על נורמת־2 של מטריצה ממשית מלבנית כלשהי. אנו מראים גם שימושים חדשים של תורת התמיכה בגרפיקה ממוחשבת ובפתרון מערכות ליניאריות סינגולריות.

בתזה זאת ארבעה נושאים עיקריים:

1. רוחב פקטור: אנו מגדירים מושג מטריציוני בשם רוחב פקטור (factor width), מדד למורכבות הקומבינטורית של מטריצות סימטריות חיוביות למחצה (מטריצות symmetric positive semi-definite, או SPSD). מושג זה נותן לנו היררכיה של מחלקות מטריצות עבור מטריצות SPSD. מטריצות עם רוחב פקטור 2 הן מטריצות שניתן לייצגן כגרפים. מטריצות עם רוחב פקטור 3 הן מטריצות שניתן לייצגן כהיפר־גרפים, כך שכל היפר־קשת מחברת לכל היותר שלושה קודקודים. באופן כללי, מטריצות עם רוחב פקטור k הן מטריצות שניתן לייצגן כהיפר־גרפים, כך שכל היפר־קשת מחברת לכל היותר k קודקודים. אנו מוכיחים כי קבוצת המטריצות הסימטריות עם רוחב פקטור 2 היאזה בדיוק קבוצת מטריצות־H עם איברים אלכסוניים אי־חיוביים. אנו מוכיחים חסמים על רוחב הפקטור, כולל חסם אחד שהוא הדוק עבור רוחבי פקטור לכל היותר 2. תוצאות אלו פורסמו במאמר "On Factor Width and Symmetric H-matrices", **Linear Algebra and its Applications** 405: 239–248 (2005) מאת Erik G. Boman, Doron Chen, Ojas Parekh and Sivan Toledo.

2. מרחבי גרעין של מטריצות־H סימטריות: אנו מאפיינים את מבנה מרחבי־הגרעין של מטריצות עם רוחב פקטור 2. אנו מראים שהמבנה של מרחב הגרעין של מטריצה עם רוחב פקטור 2 תלוי במבנה של רכיבי הקשירות של הגרף המתאים למטריצה. כל רכיב קשירות תורם לכל היותר ווקטור אחד לגרעין. אנו מספקים אפיון קומבינטורי של הדרגה של כל רכיב קשירות, ואפיון קומבינטורי של הווקטור שרכיב־הקשירות תורם לגרעין, אם ווקטור כזה קיים. עבור מטריצות סימטריות שולטות־אלכסונית (מטריצות symmetric diagonally-dominant, או SDD), אנו מציגים גם אלגוריתם קומבינטורי יעיל לבניית בסיס אורתונורמלי עבור מרחב הגרעין. אנו מראים קשר הדוק בין gain graphs לבין מטריצות־H סימטריות עם איברים אלכסוניים אי־חיוביים, שמכליל תוצאות ידועות הנוגעות לקשר בין גרפים לא מכוונים לבין מטריצות לפלסיאן, ולקשר בין signed graphs למטריצות SDD. אנו מראים כיצד ניתן לנצל אלגוריתמים קומבינטוריים אלה על־מנת לפתור מערכות ליניאריות סינגולריות מסוימות באריתמטיקת דיוק סופי (finite-precision arithmetic). תוצאות אלו פורסמו במאמר "Combinatorial Characterization of the Null Spaces of Symmetric H-Matrices", **Linear Algebra and its Applications 392: 71–90 (2004)** מאת Doron Chen and Sivan Toledo.

3. חסמים על נורמת־2 של מטריצה: אנו מראים כיצד ניתן להשתמש בשני כלים של תורת התמיכה, ה - splitting lemma וה - symmetric-product-support lemma, כדי להשיג חסמים חדשים על נורמת־2 של מטריצה ממשית. אנו השגנו שישה חסמים אלגבריים שונים וניתחנו את המשמעות

TEL AVIV UNIVERSITY אוניברסיטת תל-אביב

הפקולטה למדעים מדויקים ע״ש ריימונד ובברלי סאקלר
בית-הספר למדעי המחשב

יישומים אלגבריים ואלגוריתמיים של תורת התמיכה

חיבור לשם קבלת תואר ״דוקטור לפילוסופיה״
מאת

דורון חן

עבודה זאת נעשתה בהדרכתו של
פרופ. סיון טולדו

הוגש לסנאט של אוניברסיטת תל-אביב
ספטמבר 2005