Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Linear Algebra and its Applications 392 (2004) 71–90
LINEAR ALGEBRA AND ITS APPLICATIONS
www.elsevier.com/locate/laa

Combinatorial characterization of the null spaces of symmetric H-matrices☆

Doron Chen, Sivan Toledo *

Tel-Aviv University, School of Computer Science, Tel-Aviv 69978, Israel

Received 22 January 2004; accepted 30 May 2004

Submitted by S. Friedland

Abstract

We characterize the structure of null spaces of symmetric diagonally dominant (SDD) matrices and symmetric H-matrices with non-negative diagonal entries. We show that the structure of the null space of a symmetric SDD matrix or H-matrix A depends on the structure of the connected components of its underlying graph. Each connected component contributes at most one vector to the null space. This paper provides a combinatorial characterization of the rank of each connected component, and a combinatorial characterization of a null vector if one exists. For SDD matrices, we also present an efficient combinatorial algorithm for constructing an orthonormal basis for the null space.

The paper also shows a close connection between gain graphs and H-matrices, 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.

AMS classification: 05C22; 05C50; 65F10; 65F15

Keywords: Combinatorial matrix theory; Signed graphs; Gain graphs; Factor width; Null space; Matroids; Singular linear systems

☆This research was supported in part by an IBM Faculty Partnership Award, by grant 572/00 from the Israel Science Foundation (founded by the Israel Academy of Sciences and Humanities), and by grant 2002261 from the United-States-Israel Binational Science Foundation.

* Corresponding author.

E-mail addresses: stoledo@tau.ac.il (S. Toledo), mycroft@tau.ac.il (D. Chen).

0024-3795/$ - see front matter
doi:10.1016/j.laa.2004.05.016

1. Introduction

This paper provides a combinatorial characterization of the null spaces of three families of symmetric matrices: Symmetric diagonally dominant [1]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 [5]. A matrix A is a factor-width-2 matrix if it can be represented as $A = UU^{\mathrm{T}}$, 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. There are no known efficient algorithms for computing a width-2 factor for a general H^+ matrix, even though a factor always exists.

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 [26]). 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 [26]; these were previously called voltage graphs [8,9]).

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

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

2. Width-2 factorization of H^+ matrices

This paper analyzes the null spaces of H^+ matrices, given their so called *factor-width*-2 representation. In order to define the term *factor-width*-2, we begin with a more general definition:

Definition 2.1. We say that a real symmetric matrix A has factor-width-k, if there exists a rectangular matrix U such that $A = UU^{\mathrm{T}}$ and each column of U contains at most k non-zeros. If, in addition, there is no V with fewer than k non-zeros per column such that $A = VV^{\mathrm{T}}$we say that the factor width is *exactly* k.

In particular, we say that a real symmetric matrix A has factor-width-2, if there exists a rectangular matrix U such that $A = UU^{\mathrm{T}}$ and each column of U contains at most two non-zeros. Factor-width-2 matrices generalize SDD and SDDM matrices: it is a well-known result that an SDD matrix A can be factored into $A = UU^{\mathrm{T}}$ such that the columns of A are scaled edge vectors (either positive or negative) or half-arc vectors, defined as follows:

Definition 2.2. 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 non-zero, $\langle i\rangle_i = 1$. 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) [5]. These matrices occur frequently in engineering and scientific computation [3,19]. 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 non-singular H-matrices (see for instance, [2, Lemma 6.4]) is that they are generalized diagonally dominant, defined as follows:

Definition 2.3. A square matrix A is *generalized diagonally dominant* if there is a positive vector $y > 0$ such that for every row i,

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

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 $M(A)y \geqslant 0$, $y \geqslant 1$, 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 [17–19], 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 [22,26], 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. Gain graphs and factor-width-2 matrices

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. In this paper 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" [26].

Gain graphs (previously called voltage graphs [8,9]) are a generalization of signed graphs [26]. A signed graph is a gain graph in which all the gains are ± 1. The notion of balance was introduced by Harary [13].

Gain graphs, like undirected unweighted graphs and like signed graphs, induce a matroid. This was stated without a proof in [26]. 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 non-zeros. 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 non-zero 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 ith position and β in the jth 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 3.1. 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 non-zero, $\langle i\rangle_i = 1$.

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

Lemma 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. *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 3.4. *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 3.5. *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 $\prod_{j=1}^{k} \alpha_j = 1$.

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 $\det(A) = 1 - \prod_{j=1}^{k} \alpha_j$. □

The previous lemma is a generalization of a result by Grossman, Kulkarni, and Schochetman [11, Theorem 2.1]. They analyzed the case were all the edge vectors have gain -1.

Lemma 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 3.7. *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) of 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 3.6, 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 3.6, 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. □

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 [10] 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 $A = UU^{\mathrm{T}}$, such that U has at most two non-zeros 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 $(1, 1, 1)^{\mathrm{T}}(1, 1, 1)$ does not belong to this class.

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

Theorem 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 4.1 follows from the following lemmas.

Lemma 4.2. *Let $A_i = U_i U_i^{\mathrm{T}}$ 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 that for each vertex j in the connected component, U_i spans $\langle 1, j, \alpha \rangle$ for some $\alpha \neq 0$. By Lemma 3.4, 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. □

Lemma 4.3. *Let $A_i = U_i U_i^{\mathrm{T}}$ 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 3.6, 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 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 3.4). 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 4.4. *Let $A_i = U_i U_i^{\mathrm{T}}$ 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 components.*

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 jth 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: $u^{\mathrm{T}} y_i = \overrightarrow{0}$ for each $u \in U_i$. Therefore: $uu^{\mathrm{T}} y_i = \overrightarrow{0}$ for each $u \in U_i$. Hence,

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

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, $u^{\mathrm{T}} y_i = 0$.

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. □

We have shown that the null space of matrices $U_i U_i^{\mathrm{T}}$ 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 jth 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 non-zeros/edges). This factorization contains only edges with gain ± 1 (the corresponding edge vectors were denoted by $\langle ij \rangle$ and $\rangle ij \langle$ in [4]) and half-arcs. The gain graph corresponding to A is called a *signed graph* [26], 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 non-positive 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, [6]). 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.

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 4.2–4.4, 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.

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 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 non-zeros 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 non-zeros 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 $A = LL^{\mathrm{T}}$, 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 [15], [16, 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 $NN^{\mathrm{T}}b$. 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 [1]. 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 non-zero 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 $A\hat{x} = (I - NN^{\mathrm{T}})b$. 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 [14] and MINRES [20], 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 non-zero eigenvalues), but they converge very slowly or fail to converge on inconsistent systems. While iterative least-square solvers such as LSQR [21] 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.

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.

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 $A = UU^{\mathrm{T}}$, 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 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 non-zero structure and with the same element signs is singular. The non-zero 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 non-singular.

Therefore, if we know that the application in which the matrix arises only produces SDD matrices and that the non-zero and sign structure is accurate (not subject to numerical errors), then we can conclude that the matrix is non-singular. There is no singular SDD matrix with this non-zero 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.

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 non-zero 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)
```

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 [21], a least-squares solver, and three linear-equation solvers: Conjugate Gradients (CG) [14] (see also [7]), MINRES [20], and GMRES [23] 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 $(I - yy^{\mathrm{T}})b$ 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 paper; they were similar to the results shown.)

The results of the experiments are presented in Figs. 1–3. 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 $A\hat{x} = (I - yy^{\mathrm{T}})b$. (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 min-

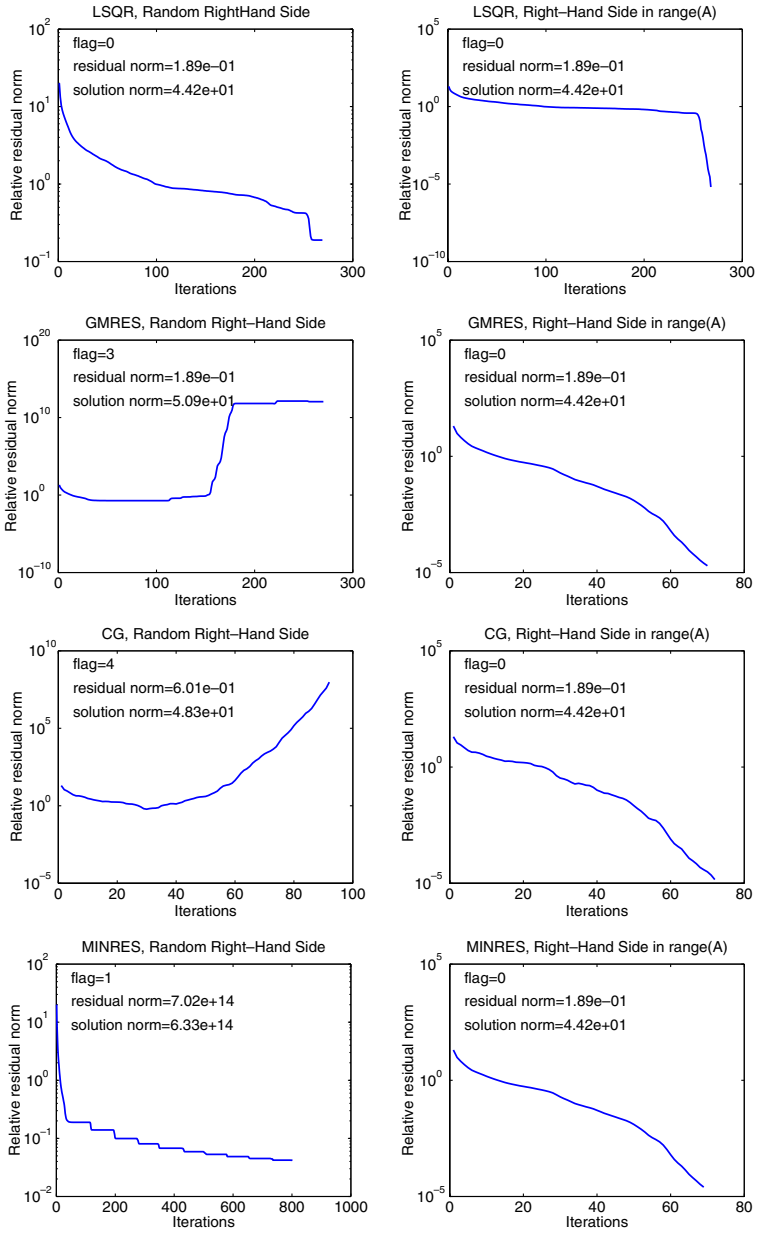

Fig. 1. The convergence of Krylov-subspace iterative solvers on a 400-by-400 singular linear system. The coefficient matrix is an SDD matrix representing a 2D mesh. The plots show (top to bottom) the convergence of LSQR, GMRES, CG, and MINRES. In each row, the plot on the left shows the convergence when 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).

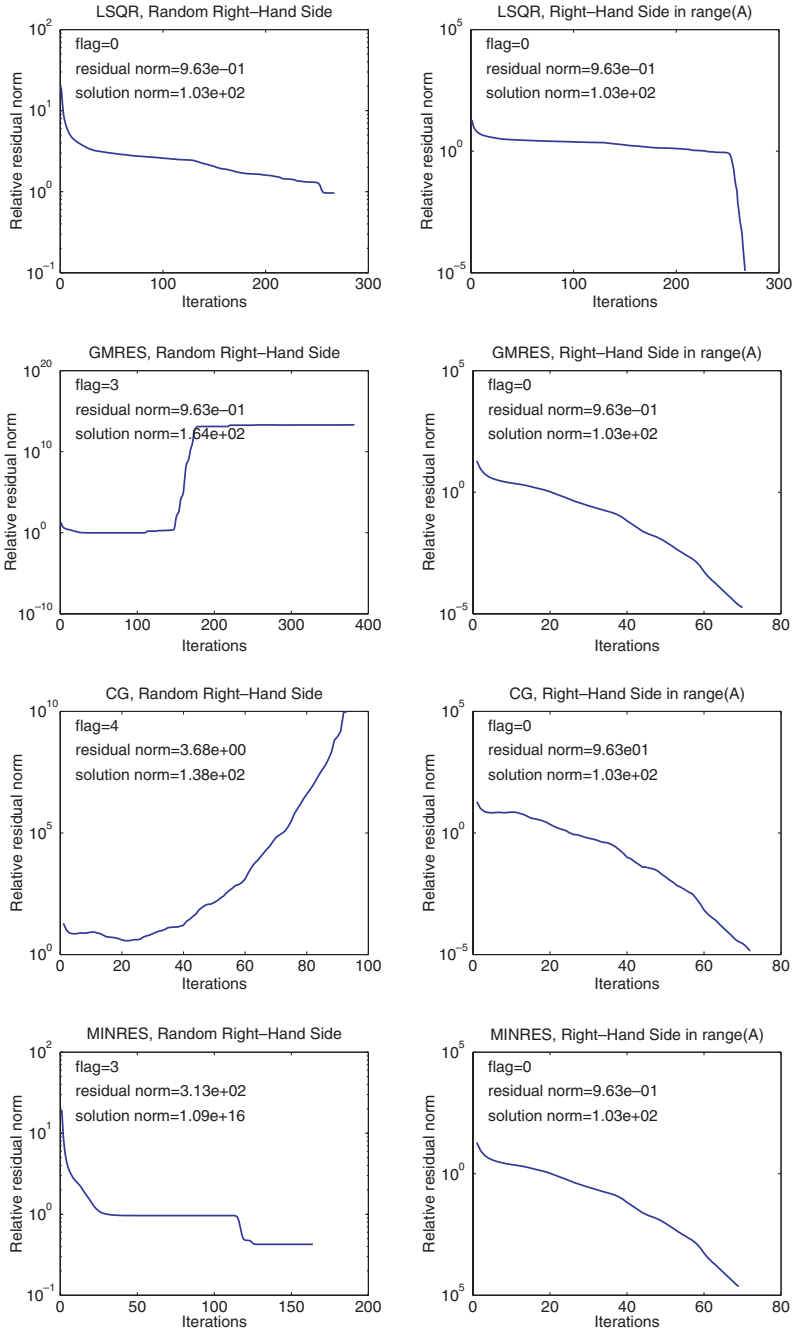

Fig. 2. An experiment on an additional right-hand-side.

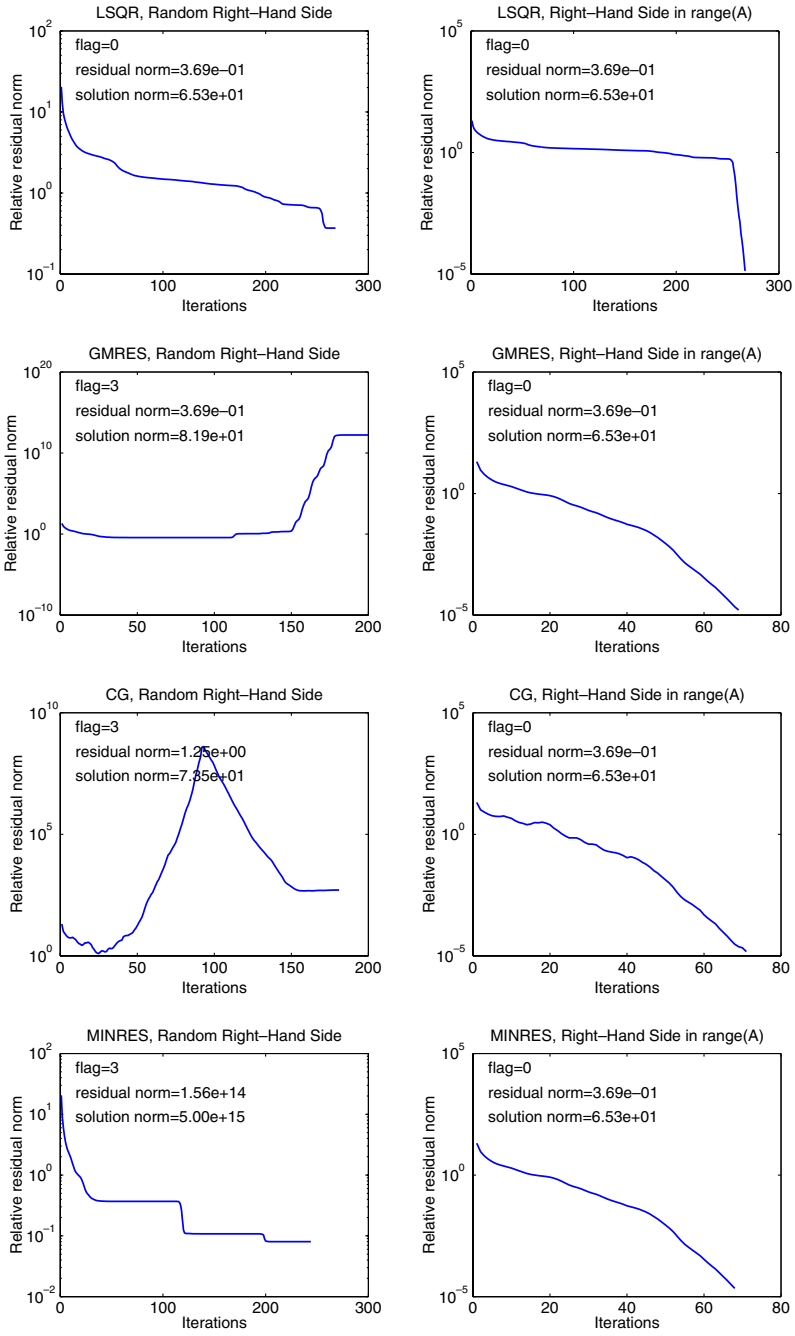

Fig. 3. An experiment on an additional right-hand-side.

imizer $\hat{x}$ 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.

8. Conclusions and open problems

The main results in this paper 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 the utility of these combinatorial algorithms in a number of important numerical-linear-algebra computations.

The paper extends previous results in spectral graph theory in several directions. First, we study spectral properties of signed and weighted 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 [12,25], but in general Laplacian eigenvectors have not been studied much.

The paper 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 [17–19], 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 [24].

Acknowledgments

Thanks to Erik G. Boman and to Ojas Parekh for numerous comments on drafts of this manuscript.

References

[1] P. Arbenz, Z. Drmač, On positive semidefinite matrices with known null space, SIAM J. Matrix Anal. Appl. 24 (1) (2002) 132–149.
[2] O. Axelsson, Iterative Solution Methods, Cambridge University Press, 1994.
[3] A. Berman, R.J. Plemmons, Nonnegative Matrices in the Mathematical Sciences, SIAM, 1994.
[4] E.G. Boman, D. Chen, B. Hendrickson, S. Toledo, Maximum weight basis preconditioners, Numer. Linear Algebra Appl., in press (doi:10.1002/nla.343).
[5] E.G. Boman, D. Chen, O. Parekh, S. Toledo, On factor width and symmetric H-matrices, submitted for publication.
[6] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press and McGraw-Hill, 1990.
[7] G.H. Golub, C.F. Van Loan, Matrix Computations, 3rd ed., Johns Hopkins University Press, 1996.
[8] J.L. Gross, Voltage graphs, Discrete Math. 9 (1974) 239–246.
[9] J.L. Gross, T.W. Tucker, Generating all graph coverings by permutation voltage assignments, Discrete Math. 18 (1977) 273–283.
[10] J.W. Grossman, D.M. Kulkarni, I.E. Schochetman, Algebraic graph theory without orientation, Linear Algebra Appl. 212/213 (1994) 289–307.
[11] J.W. Grossman, D.M. Kulkarni, I.E. Schochetman, On the minors of an incidence matrix and its Smith Normal Form, Linear Algebra Appl. 218 (1–3) (1995) 213–224.
[12] S. Guattery, On the quality of spectral separators, SIAM J. Matrix Anal. Appl. 19 (3) (1998) 701–719.
[13] F. Harary, On the notion of balance of a signed graph, Michigan Math. 2 (1953–1954) 143–146.
[14] M. Hestenes, E. Stiefel, Methods of conjugate gradients for solving linear systems, Natl. Bureau Stand. J. Res. 49 (1952) 409–436.
[15] N.J. Higham, The accuracy of floating point summation, SIAM J. Sci. Comput. 14 (4) (1993) 783–799.
[16] N.J. Higham, Accuracy and Stability of Numerical Algorithms, SIAM, 1996.
[17] T. Kohno, H. Niki, H. Sawami, Y.-M. Gao, An iterative test for H-matrix, J. Comput. Appl. Math. 115 (2000) 349–355.

[18] B. Li, L. Li, M. Harada, H. Niki, M.J. Tsatsomeros, An iterative criterion for H-matrices, Linear Algebra Appl. 271 (1998) 179–190.
[19] L. Li, On the iterative criterion for generalized diagonally dominant matrices, SIAM J. Matrix Anal. Appl. 24 (2002) 17–24.
[20] C.C. Paige, M.A. Saunders, Solution of sparse indefinite systems of linear equations, SIAM J. Numer. Anal. 12 (1975) 617–629.
[21] C.C. Paige, M.A. Saunders, Algorithm 583. LSQR: Sparse linear equations and sparse least squares, ACM Trans. Math. Software 8 (1982) 195–209.
[22] K. Rybnikov, T, Zaslavsky, Criteria for balance in abelian gain graphs with applications to geometry, submitted for publication.
[23] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (1986) 856–869.
[24] O. Sorkine, D. Cohen-Or, S. Toledo, High-pass quantization for mesh encoding, in: Proceedings of ACM/Eurographics Symposium on Geometry Processing, Aachen, Germany, 2003.
[25] D.L. Spielman, S.-H. Teng, Spectral partitioning works: Planar graphs and finite element meshes, in: IEEE Symposium on Foundations of Computer Science, 1996, pp. 96–105.
[26] T. Zaslavsky, Signed graphs, Discrete Appl. Math. 4 (1982) 47–74.