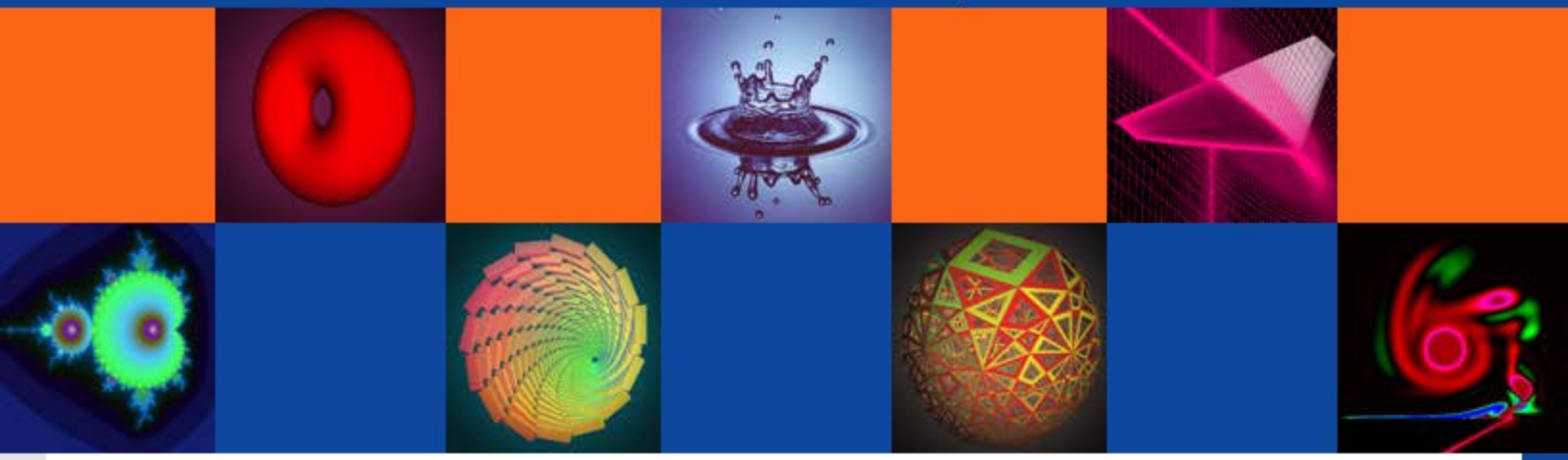
The University of Texas ARLINGTON

MATHEMATICS
Preprint Series



Self-Corrective Algorithms for Generalized Diagonally Dominant Matrices

Jinrui Guan
Linzhang Lu
Ren-Cang Li
Rongxia Shao

Technical Report 2014-18

http://www.uta.edu/math/preprint/

Self-Corrective Algorithms For Generalized Diagonally Dominant Matrices

Jinrui Guan* Linzhang Lu† Ren-Cang Li ‡ Rongxia Shao§

November 5, 2014

Abstract

A suggestive indicator is proposed for predicting whether a given (complex or real) square matrix A is or isn't a generalized diagonally dominant matrix (GDDM) by which we mean if A can be brought into a strictly diagonally dominant matrix by post-multiplying some diagonal matrix D. Based on the indicator, three self-corrective algorithms are presented for determining if A is or is not a GDDM and at the same time delivering the matrix D in case when A is a GDDM. The basic idea is to push A towards being (strictly) diagonally dominant when the indicator suggests that A is likely a GDDM or towards being off-diagonally dominant otherwise. Among the three algorithms, each has their own feature: one takes the fewest number of iterations but the most amount of work per iteration, one takes the most number of iterations but the least amount of work per iteration, and the third one strikes a balance between the two extremes. Comparing with existing methods, new algorithms are more efficient, as demonstrated on known difficult examples in the literature as well as newly designed random matrices.

Key words. Generalized diagonally dominant matrix, GDDM, M-matrix, H-matrix, self-corrective iteration

AMS subject classifications. 15B99, 65F10, 65F35

1 Introduction

A (complex or real) square matrix A is called a *generalized diagonally dominant matrix* (GDDM), also called a *nonsingular H-matrices*, if there is a diagonal matrix D such that AD is strictly diagonally dominant. GDDMs play important roles in numerical analysis, matrix theory, control systems, and mathematical economics, to name a few (see, e.g., [3, 11, 24]). In certain applications, it's critically important to know whether a given

*School of Mathematical Science, Xiamen University, P. R. China. E-mail: guanjinrui2012@163.com.

†School of Mathematics and Computer Science, Guizhou Normal University, & School of Mathematical Science, Xiamen University, P. R. China. E-mail: lzlu@xmu.edu.cn, llz@gznu.edu.cn. Supported in part by National Natural Science Foundation of China grant 10261012 and 11428104.

‡Department of Mathematics, University of Texas at Arlington, P.O. Box 19408, Arlington, TX 76019 E-mail: rcli@uta.edu. Supported in part by NSF grants DMS-1115834 and DMS-1317330, a Research Gift Grant from Intel Corporation, and NSFC grant 11428104.

§School of Mathematical Science, Xiamen University, P. R. China. E-mail: shao-rongxia@163.com.

matrix is a GDDM or not. In general this is not an easy task. In recent years, there are quite some studies on this subject, and several methods some of which are quite efficient have been obtained for determining if a given A is or isn't a GDDM and possibly a diagonal matrix D in case when it is GDDM (see, e.g., [1, 2, 4, 9, 10, 12, 14, 15, 16, 17, 18, 20]). These methods can be divided into two categories: direct methods and iterative ones. It was argued in Alanelli and Hadjidimos [1] that direct methods have high computational complexities, while iterative ones are usually much cheaper and more efficient, especially for large sparse matrices.

Among studies on iterative methods, we mention these papers [1, 2, 14, 15, 16, 17, 18, 20]. Here in this paper, our interest is also in iterative methods. Throughout this paper, $A \equiv (a_{ij})$ is always an $n \times n$ (complex or real) matrix with a_{ij} being its (i, j)th entry. Let

$$\mathbb{N} = \{1, 2, \cdots, n\},$$

and define

$$r_i(A) = \sum_{i \neq j \in \mathbb{N}} |a_{ij}|, \; t_i(A) = \frac{r_i(A)}{|a_{ii}|} \quad \text{for } i \in \mathbb{N} \tag{1.1}$$

which are the off-diagonal absolute-row sums of A and the ratios of the sums over the corresponding diagonal entries, respectively. As a convention, we let $t_i(A) = +\infty$ if $a_{ii} = 0$ (even when $r_i(A) = 0$).

For any particular i, if $t_i(A) < 1$, then the ith row is diagonally dominant, and the smaller $t_i(A)$ is, the stronger the corresponding diagonal dominance will be. If $t_i(A) < 1$ for all $1 \leq i \leq n$, i.e., $[\max_i t_i(A)] < 1$, then we say that A is strictly diagonally dominant; if, however, $[\max_i t_i(A)] \leq 1$, A is diagonally dominant. For the opposite, if $t_i(A) > 1$, then the ith row is off-diagonally dominant, and the bigger $t_i(A)$ is, the stronger the corresponding off-diagonal dominance will be. If $t_i(A) > 1$ for all $1 \leq i \leq n$, i.e., $[\min_i t_i(A)] > 1$, then we say that A is strictly off-diagonally dominant; if, however, $[\min_i t_i(A)] \geq 1$, then we say that A is off-diagonally dominant.

Often $\min_i t_i(A) \leq 1 \leq \max_i t_i(A)$ for a given matrix A. In terms of the notation $t_i(\,\cdot\,)$, A is a GDDM if and only if $\max_i t_i(AD) < 1$ for some positive diagonal D which is usually unknown and can only be obtained by nontrivial computations. In the case that A is not a GDDM, such a D doesn't exist, but it can be shown that there exists a positive diagonal D such that $\min_i t_i(AD) \geq 1$, provided A is irreducible (Theorem 2.2 below). In summary, when A is irreducible, we can always find some positive diagonal D such that either $\max_i t_i(AD) < 1$ when A is a GDDM or $\min_i t_i(AD) \geq 1$ when A is not a GMMD.

The goal of this paper is to seek a positive diagonal D such that either $\min_i t_i(AD) \geq 1$ or $\max_i t_i(AD) < 1$ (or just less than or equal to 1). This is a realizable goal for a GDDM A and an irreducible non-GDDM A, as we just mentioned in the previous paragraph. To achieve this goal, we iteratively either decrease $\max_i t_i(A)$ or increase $\min_i t_i(A)$ through updating A by post-multiplying it by some positive diagonal matrices.

But when should we work to decrease $\max_i t_i(A)$ or increase $\min_i t_i(A)$? For that, we propose to use

$$[\min_i t_i(A)] \cdot [\max_i t_i(A)] \tag{1.2}$$

as an indicator as to how likely A can be transformed to AD that is diagonally dominant or off-diagonally dominant. If the product is less than or equal to 1, then we may think

that $\min_i t_i(A)$ "dominates" $\max_i t_i(A)$ and thus it is reasonable for us to predict that A tilts towards being a GDDM. In such a case, we should work to decrease $[\max_i t_i(A)]$. On the other hand, if the product is bigger than 1, then $\min_i t_i(A)$ is "dominated" by $\max_i t_i(A)$ and thus it is reasonable for us to predict that A tilts towards being off-diagonally dominant. In such a case, we should work to increase $[\min_i t_i(A)]$.

Like at any circumstance, prediction can go wrong. When it does, the conventional thinking is that any work before the prediction changes is waste of effort. Miraculously, this is not the case for our self-corrective algorithms in this paper. As our convergence analysis will show, our algorithms continue to make progress towards the final answer even during iterations under incorrect predictions. For example, suppose the indicator (1.2) goes from above 1 to under 1 and then back above 1 again. This means that we need to switch gear from pushing A towards being off-diagonally dominant, i.e., increasing $\min_i t_i(A)$, to pushing it towards being diagonally dominant, i.e., decreasing $\max_i t_i(A)$ and then to pushing it towards being off-diagonally dominant again. A natural question is that: do we degrade $\min_i t_i(A)$ during the time when we work to decrease $\max_i t_i(A)$? If we did, the iterative process could oscillate without making progress. Luckily, such a scenario doesn't happen for our algorithms. We will prove that $\min_i t_i(A)$ at the end of the step when the indicator switches back to above 1 is strictly bigger than the one right before the indicator goes under 1. The similar statement can be said for the case when the indicator goes from under 1 to above 1 and then back to under 1 again. Therefore our self-corrective algorithms always make progress regardless what the indicator says. That is the most favorable feature of our self-corrective algorithms – never stopping making progress towards the final answer.

Bringing a GDDM A to strictly diagonally dominant AD has important numerical consequences, especially for linear systems $Ax = b$ with a GDDM A. Such linear systems can be transformed into strictly diagonally dominant linear systems. With care, the latter can be solved more accurately [5, 6, 13] than not knowing they are strictly diagonally dominant systems. Symmetric diagonally dominant linear systems can be provably solved in nearly-linear time [22].

The rest of this paper is organized as follows. In section 2, we introduce some standard definitions and basic results. In section 3, we discuss two recent iterative methods that were tested to work better than other existing ones according to the literature. We propose our self-corrective algorithms in section 4 and analyze their convergence behavior in section 5. In section 6, we present numerical examples and comparison results to demonstrate the effectiveness of our algorithms. Finally, we give a few concluding remarks in section 7.

2 Preliminaries

First, we introduce a few standard definitions [3, 24]. For $A \equiv (a_{ij}) \in \mathbb{C}^{n\times n}$, the set of all $n \times n$ complex matrices, set

$$\mathbb{N}_-(A) = \{i \in \mathbb{N} : \ |a_{ii}| < r_i(A)\}, \tag{2.1a}$$

$$\mathbb{N}_0(A) = \{i \in \mathbb{N} : \ |a_{ii}| = r_i(A)\}, \tag{2.1b}$$

$$\mathbb{N}_+(A) = \{i \in \mathbb{N} : \ |a_{ii}| > r_i(A)\}. \tag{2.1c}$$

Evidently, $\mathbb{N} = \mathbb{N}_0(A) \cup \mathbb{N}_+(A) \cup \mathbb{N}_-(A)$. I_n, or simply I if its size is clear from the content, is the $n \times n$ identity matrix.

For a vector x, $\mathrm{diag}(x)$ denotes the diagonal matrix with the entries of x on its main diagonal. For a matrix B and a vector x, $B > 0$ and $x > 0$ mean that they are entrywise positive, and similarly, $B \geq 0$ and $x \geq 0$ mean that they are entrywise nonnegative.

Definition 2.1. Suppose $A \in \mathbb{C}^{n\times n}$.

1. A is *diagonally dominant* if $\mathbb{N}_0(A) \cup \mathbb{N}_+(A) = \mathbb{N}$, and *strictly diagonally dominant* if $\mathbb{N}_+(A) = \mathbb{N}$. A is *off-diagonally dominant* if $\mathbb{N}_0(A) \cup \mathbb{N}_-(A) = \mathbb{N}$, and *strictly off-diagonally dominant* if $\mathbb{N}_-(A) = \mathbb{N}$.

2. A is said to be a *generalized strictly diagonally dominant matrix* (GDDM) if there exists a positive diagonal matrix[1] D such that AD is strictly diagonally dominant.

3. A is *reducible* if there exists a $n \times n$ permutation matrix Π such that

$$\Pi A \Pi^{\mathrm{T}} = \begin{array}{c} \\ k \\ n-k \end{array} \begin{array}{c} \begin{array}{cc} k & n-k \end{array} \\ \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix} \end{array},$$

   for some $1 \leq k < n$, where Π^{T} is the transpose of Π. It is *irreducible* if it is not reducible.

4. A is an *irreducibly diagonally dominant matrix* if it is irreducible, $\mathbb{N}_0(A) \cup \mathbb{N}_+(A) = \mathbb{N}$, and $\mathbb{N}_+(A) \neq \varnothing$, i.e., $|a_{ii}| \geq r_i(A)$ for all $i \in \mathbb{N}$ and at least one of them is strict.

Definition 2.2. Suppose $A := (a_{ij}) \in \mathbb{R}^{n\times n}$, the set of all $n \times n$ real matrices.

1. A is called a *Z-matrix* if $a_{ij} \leq 0$ for $i \neq j$.

2. Any Z-matrix A can be expressed as $A = cI - B$, where $c > 0$ and $B \geq 0$. A is called an *M-matrix* if $c \geq \rho(B)$, and a *nonsingular M-matrix* if $c > \rho(B)$, where $\rho(B)$ is the spectral radius of B.

Definition 2.3. The *comparison matrix* of $A \in \mathbb{C}^{n\times n}$ is denoted by $\mathcal{M}(A) := (m_{ij})$ and defined by

$$m_{ij} = \begin{cases} |a_{ij}|, & \text{for } i = j, \\ -|a_{ij}|, & \text{for } i \neq j. \end{cases}$$

$A \in \mathbb{C}^{n\times n}$ is called an *H-matrix* if $\mathcal{M}(A)$ is an M-matrix, and a *nonsingular H-matrix* if $\mathcal{M}(A)$ is a nonsingular M-matrix.

Next, we summarize several basic results that will be useful later. They may be found in the classical books [3, 8, 24]. We prove them here for self-containedness.

Theorem 2.1. *Suppose $A \in \mathbb{C}^{n\times n}$.*

(a) *A is a GDDM if and only if AD is a GDDM for any given positive diagonal D.*

[1] The requirement that D has positive diagonal entries is not essential, but has been imposed in the literature.

(b) *A is a GDDM if and only if it is a nonsingular H-matrix.*

(c) *An irreducible diagonally dominant matrix is a GDDM.*

(d) *If A is a GDDM, then $\mathbb{N}_+(A) \neq \varnothing$, i.e., A has at least one strictly diagonally dominant row.*

(e) *Suppose A is irreducible and $\mathbb{N}_-(A) = \varnothing$. A is a GDDM if and only if $\mathbb{N}_+(A) \neq \varnothing$.*

(f) *If A is a GDDM, then $a_{ii} \neq 0$ for all $i \in \mathbb{N}$.*

Proof. Items (a) and (f) are rather obvious by definition.

For item (b), if A is a GDDM, then there exists a positive diagonal D such that AD is strictly diagonally dominant. So $\mathcal{M}(AD) = \mathcal{M}(A)D$ is strictly diagonally dominant and thus $\mathcal{M}(A)$ is a nonsingular M-matrix, or equivalently, A is a nonsingular H-matrix. On the other hand, if A is a nonsingular H-matrix, then $\mathcal{M}(A)$ is a nonsingular M-matrix. Therefore there exists an n-vector $x > 0$ such that $\mathcal{M}(A)x > 0$. It can be verified that AD is strictly diagonally dominant, where $D = \operatorname{diag}(x)$.

Item (c) holds because any irreducible diagonally dominant matrix is nonsingular [24, p.23] and thus $\mathcal{M}(A)$ is a nonsingular M-matrix. So A is a nonsingular H-matrix. Now use item (b) to conclude the proof.

For item (d), suppose that A is a GDDM, then AD is strictly diagonally dominant for some positive diagonal $D = \operatorname{diag}(d_1, \ldots, d_n)$. Then $\mathcal{M}(A)x > 0$, where $x = [d_1, \ldots, d_n]^{\mathrm{T}}$. Let d_i be the smallest entry of x. Then

$$|a_{ii}|d_i - \sum_{j \neq i} |a_{ij}|d_j > 0 \quad \Rightarrow \quad |a_{ii}| > \sum_{j \neq i} |a_{ij}|\frac{d_j}{d_i} \geq \sum_{j \neq i} |a_{ij}|.$$

That is $i \in \mathbb{N}_+(A)$.

For item (e), if $\mathbb{N}_+(A) \neq \varnothing$, then A is a GDDM by item (c). On the other hand, if A is a GDDM, then $\mathbb{N}_+(A) \neq \varnothing$ by item (d). □

Theorem 2.1(b) leads to ways for checking if a given A is a GDDM by verifying if A a nonsingular H-matrix. The basic idea is as follows. Since $\mathcal{M}(A)$ is a Z-matrix and can be written as $\mathcal{M}(A) = cI - B$, where $c > 0$ and $0 \leq B \in \mathbb{R}^{n \times n}$. By the theory for nonnegative matrices [3, 19, 24], $\rho(B)$ is an eigenvalue and it is among one of the largest in absolute value. Compute it and then check if $c > \rho(B)$ to arrive at a conclusion. But this could be a very expensive approach, e.g., in the case when B is imprimitive, B has several eigenvalues equally spaced on the circle $\{z \in \mathbb{C} : |z| = \rho(B)\}$ [19, pp.675-680] and thus simple methods like the power method usually diverge.

If we can find a positive diagonal D such that

1. $\mathbb{N}_+(AD) = \varnothing$, i.e., $\min_i t_i(AD) \geq 1$, then A is not a GDDM by Theorem 2.1(d), or
2. $\mathbb{N}_+(AD) = \mathbb{N}$, i.e., $\max_i t_i(AD) < 1$, then A is a GDDM by definition.

In the case when A is irreducible, the condition in the second item can be weakened to

$$\min_i t_i(AD) < \max_i t_i(AD) \leq 1.$$

for claiming that A is a GDDM by Theorem 2.1(c,e).

Basically, what most existing algorithms, and ours in this paper included, try to do in finding whether A is a GDDM or not is to seek a positive diagonal matrix D such that AD is either (strictly) diagonally dominant, i.e., $\max_i t_i(AD) \le 1$, or off-diagonally dominant, i.e., $\min_i t_i(AD) \ge 1$. The question is whether for any given $A \in \mathbb{C}^{n\times n}$ it is always possible to find such a positive diagonal matrix D. In the case when A is a GDDM, by definition we can find a positive diagonal matrix D such that $\max_i t_i(AD) < 1$. What happens when A is not a GDDM? Is there a positive diagonal matrix D such that $\min_i t_i(AD) \ge 1$? The answer is yes when A is irreducible, but depends when A is reducible.

Theorem 2.2. *Suppose $A \in \mathbb{C}^{n\times n}$ is irreducible and has nonzero diagonal entries. If A is not a GDDM, then there exists a positive diagonal matrix D such that $\min_i t_i(AD) \ge 1$.*

Proof. Write $\mathcal{M}(A) = cI - B$, where $c > 0$ and $0 \le B \in \mathbb{R}^{n\times n}$. B is irreducible because A is. By Perron-Frobenius theorem [3, 19], $\rho(B)$ is a simple eigenvalue of B with the associated eigenvector $x > 0$, i.e., $Bx = \rho(B)\,x$. Thus $\mathcal{M}(A)\,x = [c - \rho(B)]x$. That A is not a GDDM implies $c \le \rho(B)$. It is not difficulty to verify that $D = \mathrm{diag}(x)$ is what we need. □

That A is irreducible in general cannot be removed from the conditions of this theorem. Consider the following block diagonal matrix

$$A = \begin{bmatrix} A_1 & \\ & A_2 \end{bmatrix} \quad \begin{array}{l} \text{with irreducible GDDM } A_1, \\ \text{and irreducible non-GDDM } A_2. \end{array} \tag{2.2}$$

This A cannot be a GDDM. Otherwise there would exist a positive diagonal matrix D such that $\max_i t_i(AD) < 1$. Partitioning $D = \mathrm{diag}(D_1, D_2)$ in the same way as for A, we find $\max_i t_i(A_2D_2) < 1$ which would imply that A_2 would be a GDDM, a contradiction. So A is not a GDDM. But there exists no positive diagonal matrix D such that $\min_i t_i(AD) \ge 1$. The latter can also be proved by a contradictory argument.

With additional conditions, we can extend Theorem 2.2 a little bit further.

Theorem 2.3. *Suppose $A \in \mathbb{C}^{n\times n}$ is reducible, has nonzero diagonal entries, and admits*

$$\Pi^{\mathrm{T}} A \Pi = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1m} \\ & A_{22} & \dots & A_{2m} \\ & & \ddots & \vdots \\ & & & A_{mm} \end{pmatrix},$$

where Π is a permutation matrix, and all A_{ii} are irreducible.

1. *If A is a GDDM, then all diagonal blocks A_{ii} are GDDMs. In other words, if one of the diagonal blocks A_{ii} is not a GDDM, then A cannot be a GDDM.*
2. *There exists a positive diagonal matrix D such that $\min_i t_i(AD) > 1$ in any one of the following two cases:*
   (a) *all diagonal blocks A_{ii} are not GDDMs and all $\mathcal{M}(A_{ii})$ are nonsingular;*
   (b) *A_{mm} is not a GDDM, $\mathcal{M}(A_{mm})$ is nonsingular, and none of A_{im} for $1 \le i \le m-1$ has zero rows.*

Proof. Without loss of generality, we may simply take $\Pi = I_n$. Since each A_{ii} is a principal submatrix of A, item 1 is a corollary of [2, Lemma 2.6].

For item 2(a), by the proof of Theorem 2.2, there are positive vectors x_i such that $\mathcal{M}(A_{ii})x_i < 0$ for $1 \le i \le m$. Let $x = [x_1^{\mathrm{T}}, x_2^{\mathrm{T}}, \ldots, x_m^{\mathrm{T}}]^{\mathrm{T}}$ and $D = \mathrm{diag}(x)$. It can be verified that $\mathcal{M}(A)\, x < 0$ which implies AD is strictly off-diagonally dominant, i.e., $\min_i t_i(AD) > 1$.

For item 2(b), by the proof of Theorem 2.2, there is a positive vector x_m such that $\mathcal{M}(A_{mm})x_m < 0$. Now pick arbitrarily positive vectors x_i for $1 \le i \le m$ of sizes compatible with A_{ii}. Since $|A_{im}|\, x_m > 0$ for $1 \le i \le m-1$, we can pick $\beta > 0$, sufficiently large such that

$$\mathcal{M}(A_{ii})x_i - \sum_{j+1}^{m-1} |A_{ij}|x_j - \beta|A_{im}|\, x_m < 0 \quad \text{for } 1 \le i \le m-1,$$

where $|A_{ij}|$ is interpreted as taking entrywise absolute values. Let $x = [x_1^{\mathrm{T}}, x_2^{\mathrm{T}}, \ldots, \beta x_m^{\mathrm{T}}]^{\mathrm{T}}$ and $D = \mathrm{diag}(x)$. It can be verified that $\mathcal{M}(A)\, x < 0$ which implies $\min_i t_i(AD) > 1$. □

3 Existing algorithms

By the discussion we had in section 2, we know

1. if $\mathbb{N}_+(A) = \varnothing$, i.e., $\min_i t_i(A) \ge 1$, then A is not a GDDM, or
2. $\mathbb{N}_+(A) = \mathbb{N}$, i.e., $\max_i t_i(A) < 1$, then A is a GDDM.

In the case when A is irreducible, we only need

$$\min_i t_i(A) < \max_i t_i(A) \le 1.$$

in order to claim that A is a GDDM. But for the general case when $\min_i t_i(A) < 1 < \max_i t_i(A)$, i.e. A has some rows that are strictly diagonally dominant and some rows that are strictly off-diagonally dominant, we cannot tell immediately whether such an A is a GDDM or not by simply examining its diagonal entries and absolute-row sums $r_i(A)$ of off-diagonal entries. Some nontrivial actions must be performed on A in order to make a determination.

There are several existing iterative algorithms that were designed to do the job [1, 2, 14, 20, 23, 24]. Among them, the two algorithms in [1, 14] are perhaps the best in efficiency and robustness. In this section, we will state and briefly discuss the algorithms and later in section 6 we will compare our algorithms against them.

We remark that an H-matrix in [14] as well as in [1] was really meant a nonsingular H-matrix, or equivalently a GDDM. To be consistent with our later algorithms, in Algorithm 1 as well as other algorithms later we will always use the word "GDDM" rather than "H-matrix" as in [1, 14].

Remark 3.1. In Algorithm 1 and those algorithms below, A is constantly updated in place by positive diagonal matrices. If we denote A at entry and exit by A_{in} and A_{out}, respectively, then, except for the trivial case at line 2, $A_{\mathrm{out}} = A_{\mathrm{in}}D$, where D is the last D at line 7. In case A is declared a GDDM, the last D at line 7 also makes that AD

Algorithm 1 [14, Algorithm B]

Given $A \equiv (a_{ij}) \in \mathbb{C}^{n\times n}$, this algorithm determines if A is or isn't a GDDM.

1: $t_i = t_i(A)$ for $i \in \mathbb{N}$, and $t_{\min} = \min_i t_i$;
2: **if** $t_{\min} \geq 1$ or $a_{ii} = 0$ for some $i \in \mathbb{N}$ **then**
3: **return** that A is not a GDDM;
4: **else**
5: $D = I_n$, $t_{\max} = \max_i t_i$;
6: **while** $t_{\max} \geq 1$ and $t_{\min} < 1$ **do**
7: $D_1 = \mathrm{diag}(t_i)$, $D = DD_1$, $A = AD_1$;
8: $t_i = t_i(A)$ for $i \in \mathbb{N}$;
9: $t_{\min} = \min_i t_i$, $t_{\max} = \max_i t_i$;
10: **end while**
11: **if** $t_{\max} < 1$ **then**
12: **return** that A is a GDDM;
13: **else if** $t_{\min} \geq 1$ **then**
14: **return** that A is not a GDDM;
15: **end if**
16: **end if**

is strictly diagonally dominant. In case A is declared not a GDDM, either A has a zero diagonal entry or A is off-diagonally dominant and the last D at line 7 makes that AD is off-diagonally dominant. It is possible that the **while**-loop may be a dead loop for some input, i.e., its loop condition cannot be unsatisfied, e.g., for the matrix (2.2). So in actual implementation, one may set a maximum allowed number of the loop iterations to avoid a dead loop situation. This last comment applies to all the algorithms in what follows.

Theorem 3.1 ([14])**.** *Suppose Algorithm* 1 *terminates after a finite number of iterations. Then*

1. *A is a GDDM if $t_{\max} < 1$;*
2. *A is not a GDDM if $t_{\min} \geq 1$.*

In 2002, L. Li [16] proposed a method for determining whether a given matrix is a GDDM or not. In 2003, Ojiro, Niki, and Usui [20] gave another method for the same task. But in 2006, Alanelli and Hadjidimos [1] came up with three counterexamples. The method in [16] fails on two of them while and the algorithm in [20] fails on the third. Specifically, the algorithm in [16] does not converge for

$$A = \begin{bmatrix} 1 & 0 & -0.5 \\ -0.5 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 & -0.5 \\ -2 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix}, \tag{3.1}$$

while the algorithm in [20] fails for

$$C = \begin{bmatrix} 1 & -2 & -1 & 0 \\ -2 & 1 & 0 & -1 \\ 0 & -1/4 & 1 & -1/2 \\ -1/4 & 0 & -1/2 & 1 \end{bmatrix}. \tag{3.2}$$

It is easy to see $A = \mathcal{M}(A) = I - \begin{bmatrix} 0 & 0 & 0.5 \\ 0.5 & 0 & 0 \\ 0 & 2 & 0 \end{bmatrix} =: I - E$. It can be seen that E is irreducible and has three eigenvalues $\rho(E)\exp(2\pi i\iota/3)$ for $i = 0, 1, 2$, all with the equal absolute value $\rho(E) = .7937$, where ι is the imaginary unit. The power method on E will diverge unless it starts with an eigenvector. The same can be said about the matrix B here.

Later in section 6 we will revisit these three matrices. Having exposed the shortcomings of these two algorithms, Alanelli and Hadjidimos [1] proposed the following Algorithm 2, intended to improve them.

Algorithm 2 [1, Algorithm $\mathbb{AH}$]

Given $A \equiv (a_{ij}) \in \mathbb{C}^{n\times n}$, this algorithm determines if A is or isn't a GDDM.

1: $A = [\text{diag}(A)]^{-1}A$ and $r_{\min} = \min_{i\in\mathbb{N}} r_i(A)$;
2: **if** $r_{\min} \geq 1$ or $a_{ii} = 0$ for some $i \in \mathbb{N}$ **then**
3: **return** that A is not a GDDM;
4: **else**
5: $D = I_n$, $r_{\max} = \max_{i\in\mathbb{N}} r_i(A)$;
6: **while** $r_{\min} \leq 1$ and $r_{\max} \geq 1$ and $r_{\min} \neq r_{\max}$ **do**
7: $D_1 = \text{diag}(\frac{1+r_1(A)}{1+r_{\max}}, \ldots, \frac{1+r_n(A)}{1+r_{\max}})$;
8: $D = DD_1$, $A = D_1^{-1}AD_1$;
9: $r_{\min} = \min_{i\in\mathbb{N}} r_i(A)$, $r_{\max} = \max_{i\in\mathbb{N}} r_i(A)$;
10: **end while**
11: **if** $r_{\min} > 1$ **then**
12: **return** that A is not a GDDM;
13: **else if** $r_{\max} < 1$ **then**
14: **return** that A is a GDDM;
15: **else if** $r_{\max} = r_{\min}$ **then**
16: **return** that $\mathcal{M}(A)$ is singular and thus A is not a GDDM;
17: **end if**
18: **end if**

Alanelli and Hadjidimos [1] also established Theorem 3.2 Algorithm 2 which is essentially the power method.

Theorem 3.2 ([1])**.** *Suppose $A \in \mathbb{C}^{n\times n}$ is irreducible.*

1. *Algorithm* 2 *always terminates, i.e., the* **while***-loop from line* 6 *to line* 10 *will exit, except possibly when* $\det(\mathcal{M}(A)) = 0$*;*

2. *If Algorithm* 2 *terminates, then its answer is correct.*

Later in [2], Alanelli and Hadjidimos presented a two-stage approach to improve Algorithm 2 for reducible A, basing on the fact that *$A \in \mathbb{C}^{n\times n}$ is not a GDDM if and only if there exists at least one principal submatrix of A that is not a GDDM* [2, Lemma 2.6]. The basic idea is to run Algorithm 2 for up to certain number of the **while**-loop iterations, and if no determination can be made at the end, then it extracts out the principal

submatrix corresponding to indices $\{i : r_i(A) \geq 1\}$ and check if the principal submatrix is not a GDDM (by running the algorithm on the extracted principal submatrix). If it is not, then A is not a GDDM (otherwise still no determination can be made). Presumably this two-stage approach can help for some difficult irreducible non-GDDMs, too.

4 New algorithms

In this section, we propose three new algorithms. Theoretic convergence analysis to support the algorithms will be presented in the next section.

Recall our discussion in the first paragraph of section 3. Our intuitive idea is to "correct" A iteratively, towards either diagonal dominance or off-diagonal dominance. To implement this idea, we will propose a measure as a suggestive indicator to suggest when the underlying matrix A can likely be made towards being diagonally dominant or off-diagonally dominant.

By how $t_i(A)$ is defined, it is quite natural to interpret $t_i(A)$ as something that quantifies the off-diagonal dominance in the ith row or, equivalently, $1/t_i(A)$ as something that quantifies the diagonal dominance of the row. Let

$$p = \arg\min_{i\in\mathbb{N}} t_i(A), \quad q = \arg\max_{i\in\mathbb{N}} t_i(A). \tag{4.1}$$

Along the same line, $t_p(A)$ and $t_q(A)$ quantify the least and the most off-diagonal dominance among all n rows. Our indicator measure is then simply the product

$$t_p t_q. \tag{4.2}$$

When $t_p t_q \leq 1$, it is decided that A tilts towards being a generalized diagonally dominant matrix, and as a result we will suppress the off-diagonal dominance of all or some off-diagonally dominant rows; likewise when $t_p t_q > 1$, it is decided that A tilts towards being a generalized off-diagonally dominant matrix, and as a result we will suppress the diagonal dominance of all or some off-diagonally dominant rows.

The next theorem shed lights on how suppression should be done.

Theorem 4.1. *Suppose $A \equiv (a_{ij}) \in \mathbb{C}^{n\times n}$ has nonzero diagonal entries and that $\mathbb{N}_+(A) \neq \varnothing$ and $\mathbb{N}_-(A) \neq \varnothing$. Let $\mathbb{J} \subset \mathbb{N}$ and set $B := AD$, where D is diagonal with diagonal entries*

$$D_{(j,j)} = \begin{cases} t_j(A), & \text{for } j \in \mathbb{J}, \\ 1, & \text{for } j \notin \mathbb{J}. \end{cases}$$

1. *If $\mathbb{J} \subseteq \mathbb{N}_+(A)$, then*

$$t_i(B) \leq \begin{cases} 1, & \text{for } i \in \mathbb{J}, \\ t_i(A), & \text{for } i \notin \mathbb{J}. \end{cases}$$

   In particular, $\max_i t_i(B) \leq \max\{1, \max_i t_i(A)\}$.

2. *If $\mathbb{J} \subseteq \mathbb{N}_-(A)$, then*

$$t_i(B) \geq \begin{cases} 1, & \text{for } i \in \mathbb{J}, \\ t_i(A), & \text{for } i \notin \mathbb{J}. \end{cases}$$

   In particular, $\min_i t_i(B) \geq \min\{1, \min_i t_i(A)\}$.

Proof. Suppose $\mathbb{J} \subseteq \mathbb{N}_+(A)$. Then for each $i \in \mathbb{J}$, $t_i(A) < 1$, and therefore for $i \in \mathbb{J}$,

$$t_i(B) = \frac{\sum_{\mathbb{J}\ni j\neq i} t_j(A)|a_{ij}| + \sum_{j\notin\mathbb{J}} |a_{ij}|}{t_i(A)|a_{ii}|} \leq \frac{\sum_{\mathbb{J}\ni j\neq i} |a_{ij}| + \sum_{j\notin\mathbb{J}} |a_{ij}|}{t_i(A)|a_{ii}|} = 1.$$

For $i \notin \mathbb{J}$,

$$t_i(B) = \frac{\sum_{j\in\mathbb{J}} t_j(A)|a_{ij}| + \sum_{\mathbb{J}\not\ni j\neq i} |a_{ij}|}{|a_{ii}|} \leq \frac{\sum_{j\in\mathbb{J}} |a_{ij}| + \sum_{\mathbb{J}\not\ni j\neq i} |a_{ij}|}{|a_{ii}|} = t_i(A). \tag{4.3}$$

This proves item 1. Item 2 can be proved in the same way. □

This theorem can be understood in this way. In the case of item 1, the off-diagonal dominance for all rows $j \in \mathbb{J} \subseteq \mathbb{N}_+(A)$ is suppressed, or equivalently the diagonal dominance for these rows are increased, although possibly not strictly (examining (4.3) for conditions for strict inequality there). In the case of item 2, the opposite outcome is achieved. Thus together with the indicator measure (4.2), we propose our Self-Corrective Iteration (SCI) detailed in Algorithm 3 for determining whether a given square matrix is a GDDM or not and, as by-product, yielding a diagonal matrix D by which AD is diagonally dominant in the case when A is a GDDM.

Remark 4.1. A few comments on Algorithm 3 are in order.

1. The input A is required to have no rows consisting of only 0 entries, excluding the diagonal entries. This is to make sure all $t_i > 0$ and in particular, $t_p > 0$ always. This condition is not restrictive in applications. In general, we can always preprocess A by checking if A has some rows that, excluding their diagonal entries, contain only 0 entries. If there are some rows like that, we can symmetrically permute these rows to the end as

$$\Pi^{\mathrm{T}} A \Pi = \begin{array}{c} \\ m \\ n-m \end{array} \begin{array}{c} \begin{array}{cc} m & n-m \end{array} \\ \begin{bmatrix} A_{11} & A_{12} \\ 0 & \widehat{D} \end{bmatrix} \end{array},$$

   where $\widehat{D}$ is diagonal. Suppose that we are interested in solving some linear system $Ax = b$. It can be seen that the system is now degenerated into a diagonal linear system $\widehat{D}x_2 = b_2$ and another not-so-trivial one: $A_{11}x_1 = b_1 - A_{12}x_2$. The diagonal linear system is straightforwardly solved as accurate as it can be. The question is how to solve the second one accurately. As we commented before, if A_{11} can be brought into a strictly diagonally dominant matrix AD_1 for some positive diagonal D_1, then this second linear system can be solve more accurately than without transforming A_{11} to $A_{11}D_1$.

2. The arg min and arg max at lines 2, 6, 18 may not be unique. When that is the case, simply picking one will do just fine.

3. The updating formula for γ_i at line 14 can be seen from that the absolute row sum for the ith row is

$$\sum_{j\in\mathbb{J}} t_j|a_{ij}| + \sum_{j\notin\mathbb{J}} |a_{ij}| = \sum_{j\in\mathbb{J}} (t_j - 1)a_{ij} + \sum_{j=1}^{n} |a_{ij}|.$$

Algorithm 3 Self-Corrective Iteration (SCI)

Given $A \equiv (a_{ij}) \in \mathbb{C}^{n\times n}$ with, excluding the diagonal entries, no rows consisting of only 0 entries, this algorithm determines if A is or isn't a GDDM.

1: $\gamma_i = \sum_{j=1}^{n} |a_{ij}|$ and $t_i = \gamma_i/|a_{ii}| - 1$ for $i = 1, 2, \ldots, n$;
2: $p = \arg\min_{i\in\mathbb{N}} t_i$;
3: **if** $t_p \geq 1$ or $a_{ii} = 0$ for some $i \in \mathbb{N}$ **then**
4: **return** that A is not a GDDM;
5: **else**
6: $D = I_n$, $q = \arg\max_{i\in\mathbb{N}} t_i$;
7: **while** $t_p < 1 < t_q$ **do**
8: **if** $t_p \cdot t_q \leq 1$ **then**
9: $\mathbb{J} = \{i : 0 \neq t_i < 1\}$;
10: **else if** $t_p \cdot t_q > 1$ **then**
11: $\mathbb{J} = \{i : t_i > 1\}$;
12: **end if**
13: **for** each $j \in \mathbb{J}$ **do**
14: $\gamma_i = \gamma_i + (t_j - 1) \cdot |a_{ij}|$ for $i = 1, 2, \ldots, n$;
15: $D_{(j,j)} = D_{(j,j)} \cdot t_j$, $A_{(:,j)} = t_j \cdot A_{(:,j)}$;
16: **end for**
17: $t_i = \gamma_i/|a_{ii}| - 1$ for $i = 1, 2, \ldots, n$;
18: $p = \arg\min_{i\in\mathbb{N}} t_i$ and $q = \arg\max_{i\in\mathbb{N}} t_i$;
19: **end while**
20: **if** $t_p \geq 1$ **then**
21: **return** that A is not a GDDM;
22: **else if** $t_q < 1$, or $t_q \leq 1$ and A is irreducible **then**
23: **return** that A is a GDDM;
24: **else if** $t_q \leq 1$ and A is reducible **then**
25: **return** that A may or may not be a GDDM;
26: **end if**
27: **end if**

4. As before, A is constantly updated in place by positive diagonal matrices. Denote A at entry and exit by A_{in} and A_{out}, respectively. Then, except for the trivial case at line 3, $A_{\text{out}} = A_{\text{in}}D$, where D is the one at exit. When Algorithm 3 claims that "A is a GDDM", A_{out} may not appear as strictly diagonally dominant in the case of an irreducible A.

5. In case when the cardinality of $\mathbb{J}$ is limited to 1, $t_j = 1$ for $j \in \mathbb{J}$ at line 17 always. The condition at line 22 can be satisfied only for $t_q = 1$ and irreducible A.

6. If line 25 is executed, nd then at the time we must have $t_q = 1$ and that A is reducible. A_{out} is diagonally dominant but may not be strictly, and A may or may

not be a GDDM. This can be seen by running Algorithm 3 with input A being

$$\begin{bmatrix} 3 & 0 & 1 \\ 0 & 2 & 1 \\ 0 & 2 & 2 \end{bmatrix}, \quad \text{or} \quad \begin{bmatrix} 3 & 0 & 1 \\ 0 & 2 & 2 \\ 0 & 2 & 2 \end{bmatrix}$$

separately. The first one is a GDDM (it can be seen by multiplying the second column by 3/4 for example) and the second one is no GDDM (since it is singular). Thus further work is needed to make a definitive determination, e.g., by verifying if $\mathcal{M}(A)$ is a nonsingular M-matrix or not along the lines we discussed after Theorem 2.1.

7. Conceivably and naturally, the two-stage approach in [2], as outlined at the end of section 3, can be adopted here to improve this algorithm (and its variations to be presented) for difficult reducible non-GDDMs such as the one in (2.2). Since the adoption is rather straightforward, we will not dwell on it in this paper but leave it to the reader.

The cost per iterative step of Algorithm 3 varies and depends on the cardinality of $\mathbb{J}$ for each particular step. The larger the cardinality is, the bigger the cost will be. We now introduce two variations of the algorithm, aiming at reducing per step cost. In the first variation, we force $\mathbb{J}$ to contain just one index, i.e., p or q, depending on the situation, and thus only one column is updated per iterative step. We name the resulting algorithm as

> **Algorithm 3a.** In Algorithm 3, replace line 9 by $\mathbb{J} = \{p\}$ and line 11 by $\mathbb{J} = \{q\}$, respectively.

for future reference. In the second variation, we strive to balance out the cost and the amount of suppression at lines 13-16 per iterative step. Notice that the amount of suppression is proportional to relative difference of t_j from 1. Therefore, it is reasonable to limit $\mathbb{J}$ to contain those indices j such that t_j relatively further away from 1. With this guideline in mind, we formulate

> **Algorithm 3b.** In Algorithm 3, replace line 9 by $\mathbb{J} = \{i : t_i t_q \leq 1\}$ and line 11 by $\mathbb{J} = \{i : t_p t_i > 1\}$, respectively.

5 Convergence analysis

In the following, we present a convergence analysis of Algorithm 3. We point out that every convergence result in this section is valid for both Algorithms 3a and 3b, too. To save spaces, we will not repeat them for the two variations.

Our first theorem reminds us of the Jacobi method for the symmetric eigenvalue problem: it produces the eigen-decomposition for any 2×2 symmetric matrix [7, 21] in one rotation. Here it is shown that Algorithm 3 will tell whether a 2×2 matrix is a GDDM in at most one iterative step. Note that a reducible $A \in \mathbb{C}^{2\times 2}$ is excluded from all eligible inputs of Algorithm 3.

Theorem 5.1. *For any irreducible $A \equiv (a_{ij}) \in \mathbb{C}^{2\times 2}$ with nonzero diagonal entries, Algorithm 3 outputs a correct answer in at most one iterative step.*

Proof. Without loss of generality, we may assume that $A > 0$ and

$$t_1 = \frac{a_{12}}{a_{11}} \geq t_2 = \frac{a_{21}}{a_{22}}.$$

Thus $p = 2$ and $q = 1$. If $t_2 \geq 1$ or $t_1 \leq 1$, then Algorithm 3 will skip lines 7-19 and go straight to line 20 and give a correct answer.

Suppose that $t_1 > 1 > t_2$. Let $\tilde{t}_i$ denote the new t_i (after one iteration). Then according to $t_1 t_2 \leq 1$ or $t_1 t_2 > 1$, we have

$$\tilde{t}_1 = \frac{a_{12}a_{21}}{a_{11}a_{22}} = t_1 t_2 \leq 1, \ \tilde{t}_2 = 1, \quad \text{or} \tag{5.1a}$$

$$\tilde{t}_1 = 1, \qquad \tilde{t}_2 = \frac{a_{12}a_{21}}{a_{11}a_{22}} = t_1 t_2 \geq 1. \tag{5.1b}$$

Thus Algorithm 3 will stop with a correct answer: A is a GDDM in the case of (5.1a) or A is not a GDDM in the case of (5.1b). □

For the ease of presentation, we introduce a new set of notations to trace the iterations in the algorithm. $A^{(0)}$ is the input A-matrix and $D^{(0)} = I_n$, and $A^{(k)}$ and $D^{(k)}$ is the A- and D-matrix in the k iterative loop after executing the **for**-loop at lines 13–16. Define, accordingly, $t_i^{(k)}$ for $1 \leq i \leq n$, and p_k and q_k with p_0 being the p at line 2 and q_0 the q at line 6.

Theorem 5.2. *If Algorithm* 3 *terminates*[2]*, then its output is correct.*

Proof. Suppose Algorithm 3 exits after m iterative steps. There are three possibilities: 1) A is not a GDDM, 2) A is a GDDM, and 3) A may or may not be a GDDM.

1. If that A is not a GDDM is claimed, then possible exits are at line 4 and line 21. If it is at line 4, then $A^{(0)} = A$ has either a 0 diagonal entry or $\mathbb{N}_+(A) = \varnothing$. By Theorem 2.1(d,f), A is not a GDDM. If it is at line 21, then we have $t_{p_m} \geq 1$, which means $\mathbb{N}_+(A^{(m)}) = \varnothing$. By Theorem 2.1(a,d), $A^{(m)}$ and thus A is not a GDDM.

2. Suppose that A is a GDDM is claimed. Then the only possible exit is at line 23. We have two cases: $t_{q_m} < 1$ or $t_{q_m} = 1$.

   If $t_{q_m} < 1$, then for all $i \in \mathbb{N}$, $t_i < 1$, i.e., $A^{(m)}$ is strictly diagonally dominant. So $A^{(m)}$ and thus A is a GDDM by Theorem 2.1(a).

   If $t_{q_m} = 1$, then A is diagonally dominant. Since the case $t_{p_m} \geq 1$ has been treated earlier at line 21, we must have $t_{p_m} < 1$ when line 23 is reached. This means $\mathbb{N}_+(A^{(m)}) \neq \varnothing$. Since the input A is irreducible, $A^{(m)}$ is also irreducible. Thus $A^{(m)}$ is an irreducible diagonally dominant matrix, and so it is a GDDM by Theorem 2.1(c) and so is A by Theorem 2.1(a).

3. The third possibility comes from line 25 of Algorithm 3. See the last comment in Remark 4.1.

This completes the proof. □

[2]If it terminates, it terminates in a finite number of iterative steps.

The next two theorems are about the "monotonicity" property of the t_p- and t_q-sequence generated by Algorithm 3. Note that the opposite to the condition of either $t_{q_k}^{(k)} \geq 1$ in item 1 of Theorem 5.3 or $t_{p_k}^{(k)} \leq 1$ in item 2 of Theorem 5.3 leads to immediate termination of the algorithm, and thus is welcome and not treated in the theorem.

Theorem 5.3 is in fact a simple corollary of Theorem 4.1.

Theorem 5.3. *In Algorithm* 3, *suppose* $t_{p_{k-1}}^{(k-1)} < 1 < t_{q_{k-1}}^{(k-1)}$.

1. *If* $t_{p_{k-1}}^{(k-1)} t_{q_{k-1}}^{(k-1)} \leq 1$ *and if* $t_{q_k}^{(k)} \geq 1$, *then* $t_{q_{k-1}}^{(k-1)} \geq t_{q_k}^{(k)}$;

2. *If* $t_{p_{k-1}}^{(k-1)} t_{q_{k-1}}^{(k-1)} > 1$ *and if* $t_{p_k}^{(k)} \leq 1$, *then* $t_{p_{k-1}}^{(k-1)} \leq t_{p_k}^{(k)}$.

This theorem implies that if $t_{p_i}^{(i)} t_{q_i}^{(i)} \leq 1$ during many consecutive iterations, then the corresponding $t_{q_i}^{(i)}$ is monotonically decreasing (possibly not strictly though). Likewise, if $t_{p_i}^{(i)} t_{q_i}^{(i)} > 1$ during many consecutive iterations, then the corresponding $t_{p_i}^{(i)}$ is monotonically increasing (again possibly not strictly).

Previously, we have mentioned that we would use $t_p t_q$ as the indicator for suggesting how likely the input matrix A can be made towards being diagonally dominant or off-diagonally dominant, and accordingly we should suppress the diagonal or off-diagonal dominance of selected rows. But this indicator is only an indicator and it may predict incorrectly. When that happens, i.e., after certain number of iterations, $t_p t_q$ may go from above (below) 1 to below (above) 1. Experience often tells us that any wrong prediction usually comes with waste of efforts in almost any circumstance. Luckily, what we have here is a counterexample to this lesson of experience. For example, suppose $t_{p_i}^{(i)} t_{q_i}^{(i)} \leq 1$ during many consecutive iterations and therefore during each of these iterations, A is "corrected" towards being diagonally dominant, and suppose at the end of these consecutive iterations, $t_{p_i}^{(i)} t_{q_i}^{(i)}$ becomes over above 1 and therefore A is "corrected" towards being off-diagonally dominant instead for the step. The next theorem says that A after this latest step is closer to being off-diagonally dominant than the one A before the many consecutive iterations for which $t_{p_i}^{(i)} t_{q_i}^{(i)} \leq 1$. In other words, the work that has been done during these consecutive iterations correcting A towards being diagonally dominant is not wasted, and somehow superstitiously the process knows that the indicator is making a wrong prediction during these consecutive iterations and continuously pushes A towards being off-diagonally dominant regardless.

Theorem 5.4 illustrates how t_p and t_q behave when $t_p t_q$ stays above (below) 1 and then goes below (above) 1 and then comes back to above (below) 1.

Theorem 5.4. *In Algorithm* 3, *suppose* $t_{p_i}^{(i)} < 1 < t_{q_i}^{(i)}$ *for* $k-1 \leq i \leq \ell$.

1. *If* $t_{p_{k-1}}^{(k-1)} t_{q_{k-1}}^{(k-1)} \leq 1$, $t_{p_i}^{(i)} t_{q_i}^{(i)} > 1$ *for* $k \leq i \leq \ell - 1$, *and* $t_{p_\ell}^{(\ell)} t_{q_\ell}^{(\ell)} \leq 1$, *then*

$$t_{q_{k-1}}^{(k-1)} \geq t_{q_k}^{(k)} > t_{q_\ell}^{(\ell)}; \tag{5.2}$$

2. *If* $t_{p_{k-1}}^{(k-1)} t_{q_{k-1}}^{(k-1)} > 1$, $t_{p_i}^{(i)} t_{q_i}^{(i)} \leq 1$ *for* $k \leq i \leq \ell - 1$, *and* $t_{p_\ell}^{(\ell)} t_{q_\ell}^{(\ell)} > 1$, *then*

$$t_{p_{k-1}}^{(k-1)} \leq t_{p_k}^{(k)} < t_{p_\ell}^{(\ell)}. \tag{5.3}$$

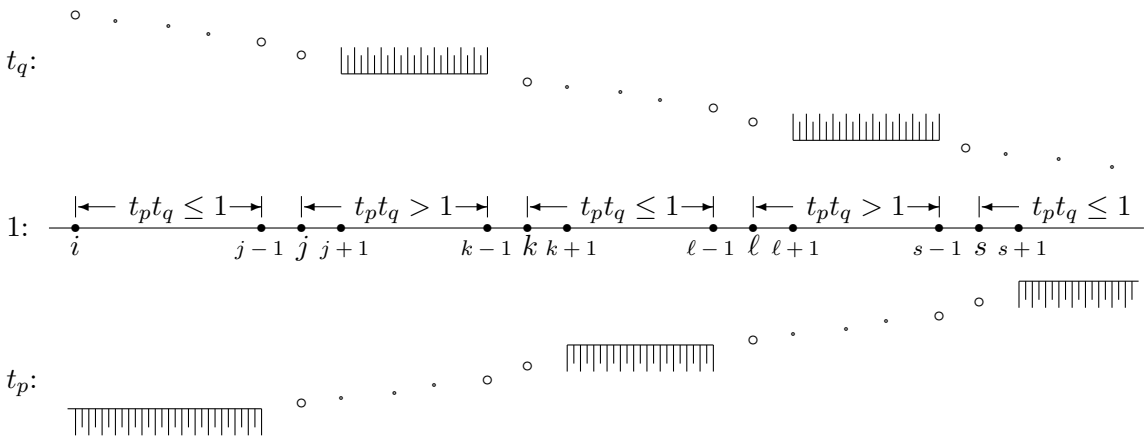


Figure 1: Behavior of t_p and t_q as $t_p t_q$ goes through region above 1 and below 1 and back. As soon as one of t_p and t_q touches 1, the iterative process stops with a claim that either A is either a GDDM or not a GDDM or a definitive determination cannot be made.

Proof. First we prove (5.2). The first inequality there is a corollary of Theorem 5.3. Also due to this theorem,

$$t_{p_k}^{(k)} \le t_{p_{k+1}}^{(k+1)} \le \cdots \le t_{p_\ell}^{(\ell)}.$$

Therefore $t_{q_k}^{(k)} > 1/t_{p_k}^{(k)} \ge \cdots \ge 1/t_{p_\ell}^{(\ell)} \ge t_{q_\ell}^{(\ell)}$, as expected.

Next we prove (5.3). The first inequality there is a corollary of Theorem 5.3, too. Also due to this theorem,

$$t_{q_k}^{(k)} \ge t_{q_{k+1}}^{(k+1)} \ge \cdots \ge t_{q_\ell}^{(\ell)}.$$

Therefore $t_{p_k}^{(k)} \le 1/t_{q_k}^{(k)} \le \cdots \le 1/t_{q_\ell}^{(\ell)} < t_{p_\ell}^{(\ell)}$, as expected. □

Combining Theorems 5.3 and 5.4, we draw Figure 1 to illustrate how t_p- and t_q-sequences move during the iterations. In Figure 1, the line at the center corresponds to 1, t_p is always below 1, and t_q is always above 1. In the shaded area with opening-up, t_q is above the bottom-line but no longer monotonic, with no upper bound, and even goes above its first value in the region ($t_p t_q > 1$). However the first value of t_q immediately after the region is guaranteed smaller than the last value of t_q right before the region. Similar statements applies to the shaded area with opening-down for t_p. The trend is that t_p moves up to 1 and t_q moves down to 1. As soon as one of them touches 1, the iterative process stops with a claim.

6 Numerical Experiments

Before we get into our numerical comparisons, we present Table 6.1 which displays the flop counts per iterative step for each algorithm we presented so far, where it is assumed that A is dense. It is clear that Algorithm 3a (for which $n_{\mathbb{J}} = 1$) has the most favorable flop counts, linearly in n, while Algorithms 1 and 2 have $O(n^2)$ flops. But since $n_{\mathbb{J}}$ changes at the runtime for Algorithms 3 and 3b, their per step cost is not known a priori. In Experiment 6.5 for random matrices, the average $n_{\mathbb{J}}$ for Algorithm 3 is about $n/2$ and

algorithm	1	2	3a	3b	3
flops	$2n^2$	$3n^2$	$n_{\mathbb{J}} \times 3n + 2n$		

Table 6.1: Flop counts per iterative step, where $n_{\mathbb{J}}$ is the cardinality of $\mathbb{J}$ and $n_{\mathbb{J}} = 1$ for Algorithm 3a always.

seems to be about 4 for Algorithm 3b on the testing matrices there. So for the experiment, Algorithm 3 costs about $1.5n^2$ per step while it is $14n$ for Algorithm 3b.

Now we present our comparison results in terms of the numbers of iterations and correctness of Algorithms 1, 2, 3, 3a, and 3b on five experiments. The first four are on "difficult" matrices in the literature, and their dimensions are between 3 and 6. The fifth experiment is for randomly generated dense and sparse matrices with dimension $n = 1000$.

Among all tests, Algorithm 3 always takes the fewest numbers of iterations. For the large random matrices in Experiment 6.5, we calculate that Algorithm 3b actually use the fewest flops overall.

Experiment 6.1. This first experiment is on two matrices A and B given in (3.1) from [1]. Our algorithms work beautifully on both matrices while the algorithm in [16] fails (see [1]). This A is also the testing matrix for Example 1 in Experiment 6.4, and B is the one for Example 2 there. All Algorithms 1, 2, 3, 3a, and 3b generate the correct claim. See Examples 1 and 2 in Table 6.2 for detail. $\diamond$

Experiment 6.2. In this experiment, we show that Algorithm 1 may malfunction, i.e., producing an erroneous claim. Consider the matrix

$$A = \begin{bmatrix} 2 & -1 & -0.5 \\ -1 & 2 & -1 \\ -1.5 & -3 & 3.5 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & -1 & 0 \\ -2 & 7 & -2 \\ -2 & -2 & 1 \end{bmatrix}.$$

A is a slight modification of [1, Example 7] in changing its $(3,3)$th entry from $-7/6$ there to 3.5 here. Test on [1, Example 7] will be reported in Table 6.2. It can be verified that this A is singular, and hence it cannot be a GDDM. But if Algorithm 1 is applied to A, at the end of 35th iteration, it is claimed that A is a GDDM. On the other hand, A is correctly identified as not a GDDM by all other algorithms. For B, all algorithms reach the same conclusion that B is not a GDDM, but Algorithms 1 and 2 take many more iterations than Algorithm 3 and its variations do, as shown in the following table

algorithm	1	2	3	3a	3b
A	-	33	31	32	31
B	28	29	3	3	3

where "-" indicates that Algorithm 1 fails to make a correct determination. $\diamond$

Experiment 6.3. This experiment is for the 6×6 matrix

$$A = \begin{bmatrix} 1 & 0.01 & 0.02 & 0.01 & 0.03 & 0.01 \\ 0.05 & 1 & 0.10 & 0.02 & 0.01 & 0.01 \\ 0.01 & 0.01 & 1 & 1.01 & 0.01 & 0.01 \\ 0.01 & 0.03 & 1.002 & 1 & 0.01 & 0.02 \\ 0.02 & 0.01 & 0.02 & 0.01 & 1 & 0.10 \\ 0.07 & 0.01 & 0.01 & 0.01 & 0.01 & 1 \end{bmatrix}$$

Example [1]	Alg. 1	Alg. 2	Alg. 3	Alg. 3a	Alg. 3b	GDDM?
1	3	4	1	2	1	Yes
2	1	4	1	1	1	No
3	1	2	1	2	1	No
4	5	6	4	11	10	Yes
5(i)	7	8	7	7	7	Yes
5(ii)	8	9	8	8	8	No
6(i)	31	32	16	30	21	Yes
6(ii)	36	37	19	26	19	No
7	-	33	32	33	32	No
8	1	2	1	1	1	Yes
9	0	0	0	0	0	No

Table 6.2: Experiment 6.4: the numbers of iterations on examples in [1, section 5]

taken from [2]. Algorithms 1 and 2 need 14 and 15 iterations, respectively, to reach the conclusion that A is not a GDDM, while Algorithms 3, 3a, and 3b take 5, 11, and 6 iterations, respectively. $\diamond$

Experiment 6.4. In the fourth experiment, we compare the numbers of iterations of the five algorithms on all the examples in [1, section 5]. The outcomes are given in Table 6.2, where the second to sixth columns starting from the second row downwards list the numbers of iterations by the respective algorithms to reach the conclusions in the seventh column[3]. Table 6.2 clearly demonstrates that Algorithm 3 uses the least numbers of iterations among all. $\diamond$

Experiment 6.5. In this experiment, we perform random testing on both dense and sparse matrices with $n = 1000$. We basically generate Z-matrices and make sure that they are either M-matrices (thus GDDMs) or not M-matrices (thus not GDDMs). Specifically, dense GDDMs are generated by, in MATLAB notation,

$$\texttt{A} = \texttt{abs}(\texttt{randn}(\texttt{n})); \texttt{s} = \texttt{max}(\texttt{abs}(\texttt{eig}(\texttt{A}))); \texttt{A} = (\texttt{s} + .01) * \texttt{eye}(\texttt{n}) - \texttt{A};$$

Change `s+.01` to `s-.01` for dense non-GDDMs. Similarly, sparse GDDMs are generated by

$$\texttt{A} = \texttt{abs}(\texttt{sprandn}(\texttt{n}, \texttt{n}, .05)); \texttt{s} = \texttt{max}(\texttt{abs}(\texttt{eig}(\texttt{full}(\texttt{A})))); \texttt{A} = (\texttt{s} + .01) * \texttt{speye}(\texttt{n}) - \texttt{A};$$

Again change `s+.01` to `s-.01` for sparse non-GDDMs. For each type, we generate 10 random matrices to test. Table 6.3 lists the average numbers of iterations of each algorithm on the 10 random matrices for each type. It clearly shows that Algorithm 3 consistently

[3]The matrix for [1, Example 8] is $A = \begin{bmatrix} \frac{1+i\sqrt{3}}{4} & 2\sqrt{2}(1+i) \\ \frac{\sqrt{2}(1-i)}{8} & 2(1-i\sqrt{3}) \end{bmatrix}$. It can be verified that $\mathcal{M}(A) = \begin{bmatrix} \frac{1}{2} & -4 \\ -\frac{1}{4} & 4 \end{bmatrix}$ which is a nonsingular matrix, but it was claimed that "$\mathcal{M}(A)$ IS SINGULAR" in [1]. One plausible explanation would be that A was misprinted in [1] since in our testing, Algorithm 2 ran correctly on this example.

A	Alg. 1	Alg. 2	Alg. 3	Alg. 3a	Alg. 3b	GDDM?
dense	13.0	14.0	11.5	3022.5	989.4	Yes
dense	13.1	14.1	11.1	3092.8	997.3	No
sparse	13.5	14.5	11.9	3564.3	1112.0	Yes
sparse	13.9	14.9	11.7	4801.8	1124.5	No

Table 6.3: Experiment 6.5: the numbers of iterations on random examples over 10 runs for each matrix type

A	Alg. 3	Alg. 3a	Alg. 3b
dense(Y)	$1.89 \cdot 10^7$	$1.51 \cdot 10^7$	$1.27 \cdot 10^7$
dense(N)	$1.75 \cdot 10^7$	$1.54 \cdot 10^7$	$1.37 \cdot 10^7$
sparse(Y)	$1.33 \cdot 10^7$	$1.51 \cdot 10^7$	$1.19 \cdot 10^7$
sparse(N)	$1.38 \cdot 10^7$	$2.03 \cdot 10^7$	$1.20 \cdot 10^7$

Table 6.4: Experiment 6.5: the average total costs in flops for each matrix type

outperforms Algorithm 1 and 2 in the numbers of iterations taken for decision-making. Since Algorithm 3 does not update all columns per iteration, unlike Algorithms 1 and 2, its per cost step, estimated at $1.5n^2$ for the dense case (average $n_{\mathbb{J}} \approx n/2$), is no higher than the latter two. Therefore Algorithm 3 uses fewest flops as well. On the other hand, the average numbers of iterations by Algorithms 3a and 3b are substantially larger than the other three algorithms. But we argue that these large average numbers of iterations can be deceptive in the sense that they do not necessarily translate into Algorithms 3a and 3b being more expensive. This is due to their low per step cost because of smaller $n_{\mathbb{J}}$ (see Table 6.1 for dense matrices). To get a sense of how small $n_{\mathbb{J}}$ can be, we list in the following table the average numbers $n_{\mathbb{J}}$ by Algorithms 3a and 3b, where "Y" and "N" indicates that the corresponding columns are for GDDMs or non-GDDMs.

algorithm	dense(Y)	dense(N)	sparse(Y)	sparse(N)
3	547.2	525.7	503.3	528.0
3b	3.6	3.9	3.9	3.9

By data in this table, Table 6.1, and Table 6.3, we can calculate the average total costs in flops for Algorithms 3, 3a, and 3b for the dense case. For the sparse case, we may estimate the average number of nonzero entries per column as $\sqrt{.05}\, n$ and thus we can use this simple model

$$(2 + \sqrt{.05}\,)n_{\mathbb{J}}\, n + 2n$$

for average flops per iterative step, where .05 is the sparse density we used in generating the sparse examples. Now we can calculate the average total costs in flops in Table 6.4. According to this table, Algorithm 3b outperforms Algorithms 3 and 3a. Moments ago, we argued that Algorithm 3 is cheaper than Algorithms 1 and 2. Putting all together, we conclude that Algorithm 3b stands out on the top in this random matrix experiment. $\diamond$

We observed that often matrices AD at exit by Algorithms 3 and 3b are strictly (off-)diagonally dominant, i.e., either $\min_i t_i(AD) > 1$ or $\max_i t_i(AD) < 1$ while those

by Algorithm 3a are just (off-)diagonally dominant , i.e., either $\min_i t_i(AD) \leq 1$ or $\max_i t_i(AD) \leq 1$ with equality.

7 Concluding remarks

We presented three self-corrective algorithms for determining if a square matrix is a generalized strictly diagonally dominant matrix or not. Although they share the same goal, by design Algorithms 3 and 3a are opposite in per step cost: the former costs more due to that all possible columns which, if updated, can help to reduce $\max_i t_i(A)$ or increase $\min_i t_i(A)$ are updated, while the latter costs less due to choosing just one significant column to update among all possible columns which, if updated, can help to achieve the same objective. Algorithm 3b strikes a balance between the two. Consequently, as we observed in numerous tests, Algorithms 3 takes the fewest iterations among our three algorithms, Algorithm 3a takes the most, and Algorithm 3b takes the number in between. Because of the differences in the per step cost, the number of iterations is no longer a good suggestion in telling which algorithm is cheaper and which is more expensive. For the random testings in Experiment 6.5, we calculated that Algorithm 3b uses the fewest flops among the three.

We compared our algorithms against two existing ones from [1, 14] – Algorithms 1 and 2. They are perhaps the most efficient and robust iterative methods in the literature. We note that per step cost by Algorithm 3 is always less than those of these two algorithms and yet in all our tests it took fewer iterations than any one of them.

Our analysis reveals that our algorithms have an elegant convergence behavior as illustrated by Figure 1. But the analysis has its unsatisfactory part: no answer is provided as to whether our algorithms will terminate in finitely many steps. We point out that all our numerical tests including those not reported here terminated successfully.

The definition of GDDM and Theorem 2.2 guarantee that the exit conditions in our algorithms are achievable, except possibly for certain reducible GDDM such as the one in (2.2). For the types of reducible GDDMs such as the ones in Theorem 2.3(2), the exit conditions can be satisfied, however. Hence extra care must be taken in dealing with difficult reducible matrices A for which the two-stage approach in [2] can be helpful (see our discussion at the end of section 3).

References

[1] M. Alanelli and A. Hadjidimos. A new iterative criterion for H-matrices. *SIAM J. Matrix Anal. Appl.*, 29(1):160–176, 2007.

[2] M. Alanelli and A. Hadjidimos. A new iterative criterion for H-matrices: the reducible case. *Linear Algebra Appl.*, 428(11-12):2761–2777, 2008.

[3] Abraham Berman and Robert J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences*. SIAM, Philadelphia, 1994. This SIAM edition is a corrected reproduction of the work first published in 1979 by Academic Press, San Diego, CA.

[4] Rafael Bru, Isabel Gimnez, and Apostolos Hadjidimos. Is $A \in \mathbb{C}^{n,n}$ a general H-matrix? *Linear Algebra Appl.*, 436(2):364–380, 2012.

[5] M. Dailey, F. Dopico, and Q. Ye. A new perturbation bound for the LDU factorization of diagonally dominant matrices. *SIAM J. Matrix Anal. Appl.*, 35(3):904–930, 2014.

[6] M. Dailey, F. Dopico, and Q. Ye. Relative perturbation theory for diagonally dominant matrices. *SIAM J. Matrix Anal. Appl.*, 2014. to appear.

[7] J. Demmel. *Applied Numerical Linear Algebra*. SIAM, Philadelphia, PA, 1997.

[8] Miroslav Fiedler. *Special Matrices and Their Applications in Numerical Mathematics*. Dover Publications, Inc., Mineola, New York, 2nd edition, 2008.

[9] Tai-Bin Gan and Ting-Zhu Huang. Simple criteria for nonsingular H-matrices. *Linear Algebra Appl.*, 374:317–326, 2003.

[10] Masunori Harada, Mastaka Usui, and Hiroshi Niki. An extension of the criteria for generalized diagonally dominant matrices. *Intern. J. Comput. Math.*, 60(1-2):115–119, 1996.

[11] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, Cambridge, 1991.

[12] Tin-Zhu Huang. A note on generalized diagonally dominant matrices. *Linear Algebra Appl.*, 225:237–242, 1995.

[13] P. Koev and F. Dopico. Perturbation theory for the LDU factorization and accurate computations for diagonally dominant matrices. *Numer. Math.*, 119:337–371, 2011.

[14] Toshiyuki Kohno, Hiroshi Niki, Hideo Sawami, and Yi-ming Gao. An iterative test for H-matrix. *J. Comput. Appl. Math.*, 115(12):349–355, 2000.

[15] Bishan Li, Lei Li, Masunori Harada, Hiroshi Niki, and Michael J. Tsatsomeros. An iterative criterion for H-matrices. *Linear Algebra Appl.*, 271(13):179–190, 1998.

[16] Lei Li. On the iterative criterion for generalized diagonally dominant matrices. *SIAM J. Matrix Anal. Appl.*, 24(1):17–24, 2002.

[17] Lei Li, Hiroshi Niki, and Morito Sasanabe. A nonparameter criterion for generalized diagonally dominant matrices. *Intern. J. Comput. Math.*, 71(2):267–275, 1999.

[18] Jianzhou Liu and Anqi He. An interleaved iterative criterion for H-matrices. *Applied Mathematics and Computation*, 186(1):727–734, 2007.

[19] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM, Philadelphia, 2000.

[20] Keiko Ojiro, Hiroshi Niki, and Masataka Usui. A new criterion for the H-matrix property. *J. Comput. Appl. Math.*, 150(2):293–302, 2003.

[21] B. N. Parlett. *The Symmetric Eigenvalue Problem*. SIAM, Philadelphia, 1998.

[22] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM J. Matrix Anal. Appl.*, 2014. to appear.

[23] Richard S. Varga. On recurring theorems on diagonal dominance. *Linear Algebra Appl.*, 13(12):1–9, 1976.

[24] R.S. Varga. *Matrix Iterative Analysis*. Springer-Verlag, Berlin, 2000.