Iterative Methods for Detecting Semipositive Matrices

Dan Hooper

Abstract

A Matrix $A \in \mathbb{R}^{n \times n}$ is said to be *semipositive* if there exists positive $x \in \mathbb{R}^n$ such that $Ax$ is positive. Semipositivity generalizes several of the notions of positivity of a matrix, including entrywise positive matrices, diagonally dominant matrices with positive diagonal elements, and P-matrices. Here, we illustrate the geometric nature of the semipositivity property, list some basic facts about semipositivity, and explore iterative methods for detecting semipositive matrices.

1: Introduction

We begin with some notation and basic facts that the reader may easily verify.

$A > 0$ will mean that $A$ is entrywise positive.

$A \geq 0$ will mean that $A$ is entrywise non-negative.

A similar convention is used to define entrywise inequality for vectors $x$.

$e = (1, 1, \ldots, 1)^T$ will denote the all ones column vector.

The set of all nonnegative vectors in $\mathbb{R}^n$ is referred to as the nonnegative orthant and denoted by $\mathbb{R}^n_+$. 

$A_i$ will denote the $i^{th}$ column of $A$.

$A^{(i)}$ will denote the $i^{th}$ row of $A$.

- A semipositive matrix must have at least one positive entry in each row.
- A matrix with a positive column is semipositive
- If $A$ is invertible, $A$ is semipositive if and only if $A^{-1}$ is semipositive.
- By continuity of $A$ considered as a map from $\mathbb{R}^n$ to $\mathbb{R}^n$, $A$ is semipositive if and only if there exists $x \geq 0$ such that $Ax > 0$.
- Diagonally dominant matrices with positive diagonal entries are semipositive.

A less obvious fact is the following.

**Theorem 1.1:** If a matrix $A$ is semipositive, then $A$ has at least one positive column sum.

**Proof:** Let $A$ be semipositive. Then there exists $x > 0$ such that $Ax > 0$. Assume, by way of contradiction, that $A$ has all negative column sums. Then $e^TA_i$ is negative for
all $i$. Now, compute
\[ e^T A x = (e^T A_1, e^T A_2, \ldots, e^T A_n) x = \sum_{i=1}^{n} e^T A_i x_i \]

On the left side, we have the dot product of two positive vectors, which gives a positive
number. On the right side, however, we are adding up positive products of the column
sums of $A$, which are all negative by assumption. This is a contradiction, and therefore at
least one of the column sums of $A$ is positive.

The reader may recall that a P-matrix is a matrix which has all of its principal minors
positive. One motivation behind the detection of semipositive matrices is to eventually
come up with an iterative method for the detection of P-matrices. In the absence of any
other information, the detection of P-matrices is a co-NP complete problem. The following
theorem gives a known relation between these two classes. The hope is, that under the
assumption that $A$ is semipositive, it will be easier to detect whether or not $A$ is a P-matrix.

The following theorem of the alternative can be found in [1].

**Theorem 1.3:** For any matrix $A$, exactly one of the following is true.

1. There exists $x \geq 0$, $x \neq 0$, such that $Ax \leq 0$.
2. There exists $y > 0$ such that $A^T y > 0$.

We will also need:

**Lemma 1.4:** A matrix $A$ is a P-matrix if and only if for each nonzero $x \in \mathbb{R}^n$, there
exists an index $j \in \{1, 2, \ldots, n\}$ such that $x_j (Ax)_j > 0$.

The proof of Lemma 1.4 is found in [3].

**Theorem 1.5:** All P-matrices are semipositive.

**Proof:**

Assume that $A$ is a P-matrix. It is then true that $A^T$ is a P-matrix. Apply Lemma 1.4 to
$A^T$ to see that alternative (1) in Theorem 1.3 with $A$ replaced by $A^T$ is false. Therefore,
alternative (2) with $A^T$ replaced by $A$ is true, so that $A$ is semipositive.

We conclude this section by giving several characterizations of semipositivity in the special
case of Z-matrices. A Z-matrix is a matrix $Z$ such that $Z_{i,j} \leq 0$ when $i \neq j$. In other
words, all of its off-diagonal entries are non-positive. It is shown in [2] that

**Theorem 1.5:** If $Z$ is a $Z$-matrix, then the following are equivalent.

- $Z$ is semipositive
- If $Z = \alpha I - B$, $B > 0$, then $\rho(B) < \alpha$
- $Re \lambda > 0$ for all eigenvalues $\lambda$ of $Z$.
- $Z$ is a nonsingular M-matrix.

**2.1: Semipositivity as an intersection of half spaces.**

Given a matrix $A \in \mathbb{R}^{n \times n}$, the set of vectors $x$ that satisfy the conditions of semipositivity is given by

$$S = \{x \mid Ax > 0, x > 0\},$$

which can also be written as

$$S = \left\{ x \left| \begin{pmatrix} A \\ I \end{pmatrix} x > 0 \right\},$$

where $I$ is the $n \times n$ identity matrix. $S$ is an open polyhedron. The question of whether or not $A$ is semipositive is thus equivalent to the question of whether or not the open polyhedron $S$ is nonempty. The $i^{th}$ row of $A$, $A^{(i)}$, defines a strict linear inequality, $A^{(i)}x > 0$. We can see that this inequality defines an open linear half-space. In order for $A$ to be semipositive, all of these half-spaces must intersect at some vector in the nonnegative orthant $\mathbb{R}_{+}^{n}$.

Example: We can see in the picture below that

$$A = \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}$$

is semipositive because the half-spaces defined by the row vectors $[2, -1]$ and $[-1, 1]$ intersect in $\mathbb{R}_{+}^{2}$. 

3
The darker (purple) region indicates the intersection of the two half-spaces. Any vector which lies in this region also lies in $S$. One can tell by looking at the picture that the vector $x = [1, 1.5]^{T}$ is in $S$. Indeed,

$$Ax = \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1.5 \end{bmatrix} = \begin{bmatrix} .5 \\ .5 \end{bmatrix} > 0$$

### 2.2: Semipositivity as an Intersection of Two Polyhedral Cones.

Given $A \in \mathbb{R}^{n \times n}$, the set

$$C = \{Ax \mid x > 0\} = \left\{ \sum_{i=1}^{n} A_{i}x_{i} \mid x > 0 \right\} = cone \{A_{i}\}$$

is also known as the open cone of the columns of $A$. To be in $C$ does not impose $Ax$ being positive. However, if we intersect $C$ with the nonnegative orthant $\mathbb{R}_{+}^{n}$, we clearly get the set $S$ from the previous section. The nonnegative orthant is also an open polyhedral cone, as it is the cone of the standard basis vectors in $\mathbb{R}^{n}$. Therefore, the semipositivity problem is a special case of determining whether or not two open polyhedral cones intersect.

### 3.1: A first attempt at an iterative algorithm

First, we notice a key fact which helps in the development of an iterative algorithm.
Theorem 3.1: A matrix $A$ is semipositive if and only if there exists a diagonal matrix $D$, with all diagonal entries positive, such that $AD$ has positive row sums.

Proof: If $A$ is semipositive, then there exists $x > 0$ such that $Ax > 0$. Let $D$ be the diagonal matrix such that $D_{ii} = x_i$. Then $De = x$. Therefore,

$$ADe = Ax > 0$$

Since $ADe$ gives the row sums of $AD$, this direction is proved. Conversely, if there exists a diagonal matrix $D$ such that $ADe > 0$, then just let $x_i = D_{ii}$ and the result follows.

Another interesting fact is the following:

Theorem 3.2: If $D$ is a diagonal matrix with $D_{ii} > 0$ for all $i$, then

$$A \text{ is semipositive } \iff AD \text{ is semipositive}$$

Proof: If $A$ is semipositive, then by theorem 3.1, $ADe > 0$. Since $e > 0$, we get that $AD$ is semipositive. Conversely, if $AD$ is semipositive there exists $x > 0$ such that $ADx > 0$. Let $\tilde{x} = Dx$. Then $\tilde{x} > 0$ and $A\tilde{x} = ADx > 0$.

What the following algorithm does is compute the row sums of $A$, and then scales the columns of $A$ by right multiplying $A$ with an appropriate diagonal matrix $D$ in an attempt to make the row sums positive. With each iteration, we construct a new $D$ which satisfies the following conditions.

1. $\min_{1 \leq i \leq n} (ADe)_i > \min_{1 \leq i \leq n} (Ae)_i$
2. $(Ae)_i > 0 \implies (ADe)_i > 0$

The first condition guarantees that the “worst” row sum of the matrix does not decrease. The second condition serves to keep positive row sums positive, so as not to destroy previous gains.

The proposed algorithm satisfies the above conditions by scaling $A$ one column at a time. In a given iteration, each column will be scaled by a positive number. First column 1 is scaled, then column 2, etc. until each column has been scaled. This process repeats as many times as is necessary. In order to satisfy the conditions proposed above, we compute a vector $M$ which determines an acceptable range of values for the current column (column $j$) to be scaled by. Here, $i_{Smallest}$ is the index of the row which currently has the smallest (most negative) row sum. In the case of $A(i_{Smallest},j) < 0$, we must scale down column $j$ because in this case this entry if the matrix is making the row sum in row $i_{Smallest}$
worse. The maximum entry of the vector $M$ gives us a lower bound on a scaling factor that will satisfy the proposed constraints above. The case is similar if $A(iSmallest,j) > 0$.

Here is the MATLAB algorithm for computing the scale factor for column $j$

```matlab
if (A(iSmallest,j)<0) %In this case we will scale down column j
disp('scale down')
    for i =1:n
        if( A(i,j) <=0 ) %then rowsum i will increase by scaling down column j
            M(i) = 0;
        elseif (rowSums(i) > 0) %trying to keep positive row sums positive after scaling.
            M(i) = 1 - rowSums(i)/A(i,j);
        else %here we are satisfying condition 1 above
            M(i) = 1 - (1/A(i,j) ) * (rowSums(i) - lowestRowSum );
        end
    end
    if (max(M) >=1) %then we can’t scale this column down
        disp('failed to scale down')
        scaleFactor = 1; %We aren’t doing anything...
    else
        scaleFactor = (1/2) * (max(M) + 1);
    end
else %It must be that A(i,j) >0, so we will scale up column j
    disp('scale up')
    for i =1:n
        if( A(i,j) >=0 ) %then rowsum i will increase by scaling up column j
            M(i) = Inf;
        elseif (rowSums(i) > 0) %trying to keep positive row sums positive after scaling.
            M(i) = 1 - rowSums(i)/A(i,j);
        else %here we are again satisfying condition 2
            M(i) = 1 - (1/A(i,j) ) * (rowSums(i) - lowestRowSum );
        end
    end
    if ( min(M) <=1) %then we can’t scale this column up
        disp('failed to scale up')
        scaleFactor = 1; %We aren’t doing anything...
    else
        scaleFactor = (1/2) * ( min(M) + 1 );
    end
end %end of computing scale factor
```
3.2: Problems With Scaling One Column at a Time

This section is devoted to explaining the testing of the algorithm, some of its interesting properties, and eventually the realization that it does not solve the problem in general. One of the first clues that the algorithm was problematic was the strange convergence properties of the matrix $D$. In the following chart, we plot the iteration number versus $\|A(D^{(k+1)} - D^{(k)})e\|_\infty$, where $D^{(k)}$ is the $k^{th}$ iteration of the diagonal matrix $D$. The chart is for a random semipositive matrix $A$.

One thing to notice is that between iterations 30 and 90, the matrix iterates $D^{(k)}$ seem to make no difference. Then, suddenly, progress is made and the matrix is found to be semipositive. But one now asks the question, “After how many iterations with no progress can we declare that a matrix is not semipositive?” In other words, we want to know good stopping criteria.

What is happening here is that condition (2) above causes the algorithm to become “stuck.” One of the row sums of $AD^{(k)}$ is positive but nearly zero after iteration 30. Small changes are being made with each iteration, until eventually (iteration 91) the row sum is zero to machine precision. At that point, condition (2) does not apply for that particular row, and so progress can be made.
The next idea we tried was dropping the second condition in the algorithm and focusing solely on improving the worst row sum from iteration to iteration. This fixed the problem of making no progress and then suddenly making progress. The following surface plot shows the iteration number, the row number, and current row sum for a randomly generated 10 × 10 semipositive matrix. One can see that as the iteration number increases, the minimum of all of the row sums tends to increase. By iteration 6, all of the row sums are positive, and hence the matrix is found to be semipositive.

Unfortunately, this doesn’t always work either. In the next figure, one can see where the problem in both algorithms lie. If two or more of the rows “tie” for the smallest row sum, then the algorithm goes back and forth trying to improve those rows without making the other rows worse. In other words, it gets stuck. In the following example, one can see that rows 1,2, and 5, are “stuck” at a row sum of approximately -40.

This behavior does not necessarily indicate that the matrix is not semipositive. After some time, doubts about whether or not the algorithm works arise. A counterexample to the validity of the algorithm would be a matrix which is semipositive, but where no improvement can be made by scaling only one column at a time. The following matrix $J$
provides that counterexample.

\[ J = \begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
2 & -1 & 1 & -3 \\
-1 & 2 & -3 & 1
\end{bmatrix} \]

Clearly \( J \) is semipositive, because

\[
\begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
2 & -1 & 1 & -3 \\
-1 & 2 & -3 & 1
\end{bmatrix}
\begin{bmatrix}
10 \\
10 \\
1 \\
1
\end{bmatrix} = \begin{bmatrix}
22 \\
22 \\
8 \\
8
\end{bmatrix}.
\]

If you start with the all ones vector (giving the row sums) there is no way to make an improvement using only one column of \( J \).

\[
\begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
2 & -1 & 1 & -3 \\
-1 & 2 & -3 & 1
\end{bmatrix}
\begin{bmatrix}
1 \\
1 \\
1 \\
1
\end{bmatrix} = \begin{bmatrix}
4 \\
4 \\
-1 \\
-1
\end{bmatrix}.
\]

One could scale up columns 1 and 2 simultaneously, or scale down columns 3 and 4 simultaneously, but there is no way to scale just one column and improve the most negative row sum. This demonstrates the failure of any algorithm which tries to scale one column at a time and at the same time get an improvement defined by condition (1) above. It is also conjectured that scaling only a proper subset of the columns at any given time can fail to produce a sufficient algorithm. A new idea is needed.

4: The Projection Method

In the field of linear programming, the “feasibility” of a system \( Mx \geq b \) has been studied in detail. A system is “feasible” if at least one solution to the system of inequalities exists. In order to take advantage of this theory in the context of semipositivity, a simple observation is needed.

**Theorem 4.1:** A matrix \( A \) is semipositive if and only if the linear inequality system

\[
M = \begin{pmatrix} A \\ I \end{pmatrix} x \geq e
\]

is feasible.

**Proof:** First, assume that the system

\[
\begin{pmatrix} A \\ I \end{pmatrix} x \geq e
\]
has a solution $x$. It follows that $x = Ix \geq e > 0$ and $Ax \geq e > 0$. Therefore, $A$ is semipositive. Conversely, assume that $A$ is semipositive, so that there exists $x > 0$ such that $Ax > 0$. Let $\alpha$ be defined by

$$\alpha = \min \left\{ \min_{1 \leq i \leq n} x_i, \min_{1 \leq i \leq n} (Ax)_i \right\}$$

Then one can see that $\alpha > 0$ and if we define $\tilde{x} = \frac{1}{\alpha} x$, then

$$\begin{pmatrix} A \\ I \end{pmatrix} \tilde{x} \geq e$$

Now, it is important to note that the vector $e$ had little to do with the proof of Theorem 4.1. In fact, we could replace $e$ with any positive vector and get an analogous result. Note that we cannot simply replace $e$ with $0$ in the above theorem, because then the zero vector would always satisfy $Mx \geq 0$, so that every matrix is semipositive, which is false. Theorem 4.1 allows us to take advantage of the projective methods for solving systems of linear inequalities, described below.

One of the earliest projection methods for solving systems of linear inequalities was proposed by Motzkin and is found in [4]. The idea is relatively simple. Take any $x^{(0)}$ as an initial “guess” for a solution of $Mx \geq b$. If $x^{(0)}$ is not a solution, then at least one entry of the vector $Mx^{(0)} - b$ is negative. In other words, $x^{(0)}$ is on the “wrong side” of at least one of the hyperplanes defined by the inequality system $Mx \geq b$. Let $x^{(1)}$ be the orthogonal projection of $x^{(0)}$ onto the furthest such hyperplane. If $x^{(1)}$ is not a solution to the system, the process repeats. In other words, $x^{(0)}$ is an arbitrary initial guess, and if $x^{(k)}$ is known, $x^{(k+1)}$ is defined by:

$$x^{(k+1)} = x^{(k)} + t M^{(i')}$$

where $M^{(i)}$ is the $i^{th}$ row of $M$, $i'$ is the index that achieves

$$\max_{1 \leq i \leq n} \frac{(b - Mx^{(k)})_i}{\|M^{(i)}\|_2}$$

and $t$ is defined by:

$$t = \frac{(b - Mx^{(k)})_{i'}}{\|M^{(i')}\|_2^2}$$

One can simplify these formulas greatly by normalizing the rows of $M$.

The next two theorems characterize the convergence properties of this algorithm, and are found in [4].
**Theorem 4.2:** Let $Mx \geq b$ be a consistent system of linear inequalities and let $\{x^{(v)}\}$ be the sequence of iterates in the algorithm defined as above. Then $x^{(v)} \rightarrow x$ where $x$ is a solution to the system. Furthermore, if $R$ is the distance of $x^{(0)}$ from the nearest solution, we have

$$|x^{(v)} - x| \leq 2R\theta^v \quad v = 0, 1, \ldots$$

where $0 < \theta < 1$ depends only on the matrix $M$.

Upper bounds for $\theta$ are known, and can also be found in [4]. They are somewhat messy, and so are omitted. An obvious next step might be to come up with some simpler upper bounds on $\theta$ in order to be able to define a “stopping point” for the proposed algorithm for detecting semipositive matrices.
References


