On Polynomial Coefficients and Rank Constraints

Anders Helmersson
Division of Automatic Control
E-mail: andersh@isy.liu.se

4th February 2009

Report no.: LiTH-ISY-R-2878

Address:
Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping, Sweden

WWW: http://www.control.isy.liu.se
Abstract
Rank constraints on matrices emerge in many automatic control applications. In this short document we discuss how to rewrite the constraint into a polynomial equations of the elements in a the matrix. If addition semidefinite matrix constraints are included, the polynomial equations can be turned into an inequality. We also briefly discuss how to implement these polynomial constraints.

Keywords: Rank constraints, characteristic polynomials
On Polynomial Coefficients and Rank Constraints

Rank constraints on matrices emerge in many automatic control applications. In this short document we discuss how to rewrite the constraint into a polynomial equations of the elements in a the matrix. If addition semidefinite matrix constraints are included, the polynomial equations can be turned into an inequality. We also briefly discuss how to implement these polynomial constraints.

Keywords
Rank constraints, characteristic polynomials
1 Introduction

We employ the coefficients in characteristic polynomial of a matrix \(Z\) as closeness measure to a rank constraint. We assume that \(Z\) has no negative eigenvalues.

Rank constraints are for instance used when searching for reduced-order \(H_\infty\) controller. The existence of such a controller can be described in terms of two linear matrix inequalities (LMIs) in two symmetric matrix variables, \(X, Y \in \mathbb{R}^{n \times n}\), where \(n\) denotes the order of the system to be controlled, see [2]. A third LMI connects \(X\) and \(Y\):

\[
\begin{bmatrix}
  X & I \\
  I & Y \\
\end{bmatrix} \succeq 0. 
\]

(1)

The existence of a controller of reduced order, \(r < n\), is given by

\[
\text{rank} \left[ \begin{bmatrix} X & I \\ I & Y \end{bmatrix} \right] \leq n + r. 
\]

(2)

The rank condition (2) can also be rewritten (using Schur complement) as

\[
\text{rank}(XY - I) \leq r. 
\]

(3)

This report is an extension of LiTH-ISY-R-2867. New upper and lower bounds of \(c_{n-r-1}/c_{n-r}\) with proofs are provided.

2 Polynomial Criterion

The characteristic polynomial of a matrix \(Z \in \mathbb{R}^{n \times n}\) is defined by

\[
\det(\lambda I - Z) = \sum_{i=0}^{n} c_i(Z)\lambda^i. 
\]

(4)

where the coefficients, \(c_i(Z)\), are polynomial functions of the elements in \(Z\). For instance, \(c_0(Z) = \det(Z)\), \(c_{n-1}(Z) = \text{tr}(Z)\) and \(c_n(Z) = 1\). Note that when \(Z = I - XY\), the characteristic polynomial can also be defined in terms of the Hankel singular values, \(\sigma_i\), of \((X, Y)\) as

\[
\det((\lambda - 1)I + \Sigma^2) = \prod_{i=1}^{n} (\lambda + \sigma_i^2 - 1) = \sum_{i=0}^{n} c_i\lambda^i.
\]

Lemma 1. Let \(Z \in \mathbb{R}^{n \times n}\) be a matrix with real non-negative eigenvalues, \(\lambda_i(Z) \geq 0\), and let \(c_i(-Z)\) be the coefficients of the characteristic polynomial of \(-Z\) as defined in (4). Then, the following statements are equivalent if \(r < n\):

(i) \(c_{n-r-1}(-Z) = 0\);

(ii) \(\text{rank } Z \leq r\).

Proof. Showing (ii) \(\Rightarrow\) (i) is trivial, since (ii) is equivalent to that \(\lambda_{r+1}(Z) = \ldots = \lambda_n(Z) = 0\), where \(\lambda_i(Z)\) denotes the \(i\)th eigenvalue of \(Z\). To show (i) \(\Rightarrow\) (ii), we note that

\[
c_{n-r-1}(-Z) = \sum_{J \in C_{r+1}(\{1,n\})} \prod_{i \in J} \lambda_i(Z), 
\]

(5)
where \( C_i(U) \) denotes the set of all combinations of \( i \) elements from the set \( U \); for instance, \( C_2(\{1, 2, 3\}) = \{(1, 2), (1, 3), (2, 3)\} \).

Since, every \( \lambda_i(Z) \geq 0 \), we conclude that \( c_{n-r+1}(-Z) = 0 \) implies that every product of (5) must be zero, and consequently at least one factor, \( \lambda_i(Z) \), in each product must be zero. Thus, at least \( n - r \) factors, \( \lambda_i(Z) \) must be zero, and consequently (ii) holds.

Instead of using the coefficient \( c_{n-r+1}(-Z) \) directly as a measure of the matrix’s closeness to the rank condition, \( \text{rank} \leq r \), we instead use \( \frac{c_{n-r+1}(-Z)}{c_{n-r}(-Z)} \) as a measure of closeness. It has the nice property that if \( n - r \) eigenvalues values, \( \lambda_{r+1}, \ldots, \lambda_n \), are close to 0 and the remaining ones are large, then

\[
\frac{c_{n-r+1}(-Z)}{c_{n-r}(-Z)} \approx \sum_{i=r+1}^{n} \lambda_i(Z).
\]

More specifically, the following relations hold in general.

**Lemma 2.** Let \( Z \) be a matrix with non-negative eigenvalues ordered by \( \lambda_1(Z) \geq \lambda_2(Z) \geq \ldots \geq \lambda_n(Z) \geq 0 \). Then, the following relations hold:

\[
\frac{1}{r+1} \sum_{i=r+1}^{n} \lambda_i(Z) \leq \frac{c_{n-r+1}(-Z)}{c_{n-r}(-Z)} \leq \sum_{i=r+1}^{n} \lambda_i(Z),
\]

or, equivalently,

\[
\frac{c_{n-r+1}(-Z)}{c_{n-r}(-Z)} \leq \sum_{i=r+1}^{n} \lambda_i(Z) \leq (r+1) \frac{c_{n-r+1}(-Z)}{c_{n-r}(-Z)}.
\]

**Proof.** The upper bound in (6) (the lower bound in (7)) is obtained by observing that \( C_r([1, k-1]) \subseteq C_r([1, n]) \) assuming \( k \leq n \), and consequently

\[
\left( \sum_{k=r+1}^{n} \lambda_k \right) \sum_{J \in C_r([1, n])} \prod_{i \in J} \lambda_i \geq \sum_{k=r+1}^{n} \left( \lambda_k \sum_{J \in C_r([1, k-1])} \prod_{i \in J} \lambda_i \right)
= \sum_{J \in C_{r+1}(1, n)} \prod_{i \in J} \lambda_i,
\]

where we have dropped the argument \((Z)\).

The lower bound in (6) (the upper bound in (7)) is obtained by first observing that

\[
(r+1) \sum_{J \in C_{r+1}(1, n)} \prod_{i \in J} \lambda_i = \sum_{J \in C_r([1, n])} \left( \sum_{k \in [1, n] \setminus J} \lambda_k \prod_{i \in J} \lambda_i \right).
\]

\footnote{Formally, we can define \( C_i(U) \) recursively by \( C_0 = \{\emptyset\} \), and \( C_{i+1}(U) = \{\{x\}\} \cup \{y : x \in U, y \in C_i(U \setminus \{x\}\} \) for \( i \geq 0 \).}
Every term in the left-hand side is generated by the right-hand side in \((r + 1)\) copies, which is the reason for the factor in the left-hand side. Next using the fact that the eigenvalues, \(\lambda_k\) are ordered we can use

\[
\sum_{k \in [1, n] \setminus J} \lambda_k \geq \sum_{k = r + 1}^{n} \lambda_k,
\]

and consequently,

\[
(r + 1) \sum_{J \in C_{r+1}([1, n])} \prod_{i \in J} \lambda_i \geq \sum_{J \in C_r([1, n])} \left( \sum_{k = r + 1}^{n} \lambda_k \prod_{i \in J} \lambda_i \right)
\]

\[
= \left( \sum_{k = r + 1}^{n} \lambda_k \right) \sum_{J \in C_r([1, n])} \prod_{i \in J} \lambda_i,
\]

from which the relation follows.

3 Derivatives of \(c_i(Z)\)

Let \(C(Z)\) denote the characteristic polynomial of \(Z(x)\) and let \(Z_i = \frac{\partial}{\partial x_i} Z(x)\):

\[
C(Z) = \det(\lambda I - Z) = \sum_{i=0}^{n} c_i(Z)\lambda^i.
\]

We can use the fact that

\[
\frac{\partial}{\partial x_i} \log \det Z(x) = \text{tr} Z^{-1}(x) Z_i.
\]

We apply this on \(C(Z)\),

\[
\frac{\partial}{\partial x_i} \det(\lambda I - Z(x)) = -\det(\lambda I - Z(x)) \text{tr}(\lambda I - Z(x))^{-1} Z_i.
\]

Next, introduce the polynomial

\[
P(\lambda) = c_n I \lambda^{n-1} + (c_n Z + c_{n-1} I) \lambda^{n-2} + \ldots + \left( c_n Z^{n-1} + c_{n-1} Z^{n-2} + \ldots + c_1 I \right).
\]

Then

\[
P(\lambda)(\lambda I - Z(x)) = c_n I \lambda^{n-1}(\lambda I - Z(x))
\]

\[
+ (c_n Z + c_{n-1} I) \lambda^{n-2}(\lambda I - Z(x))
\]

\[
+ \ldots
\]

\[
+ \left( c_n Z^{n-1} + c_{n-1} Z^{n-2} + \ldots + c_1 I \right) (\lambda I - Z(x))
\]

\[
= \left( c_n \lambda^n + c_{n-1} \lambda^{n-1} + \ldots + c_1 \lambda \right) I
\]

\[
- \left( c_n Z^n + c_{n-1} Z^{n-1} + \ldots + c_1 Z \right)
\]

\[
= \left( c_n \lambda^n + c_{n-1} \lambda^{n-1} + \ldots + c_1 \lambda + c_0 \right) I
\]

\[
= \det(\lambda I - Z(x)) I
\]
where we have used the fact that the characteristic polynomial of $Z$ applied to itself yields zero.

Consequently, $(\lambda I - Z(x))^{-1} = P(\lambda)/\det(\lambda I - Z(x))$, and

$$\frac{\partial}{\partial x_i} C(Z) = \frac{\partial}{\partial x_i} \det(\lambda I - Z(x)) = -\text{tr} P(\lambda) Z_i,$$

and

$$\frac{\partial c_k}{\partial x_i} = \begin{cases} 0, & k = n \\ -\text{tr} \left(c_n Z^{n-k-1} + c_{n-1} Z^{n-k-2} + \ldots + c_{k+1} I\right) Z_i, & \text{otherwise} \end{cases} \quad (8)$$

Note that $c_k(Z)$ can be computed as a polynomial function of $\text{tr} Z$, $\text{tr} Z^2$, $\ldots$, $\text{tr} Z^{n-k}$. For instance, $c_n = 1$, $c_{n-1} = -\text{tr} Z$, $c_{n-2} = \frac{1}{2} ((\text{tr} Z)^2 - \text{tr} Z^2)$, and $c_{n-3} = \frac{1}{6} (-2 \text{tr} Z^3 + 3(\text{tr} Z)(\text{tr} Z^2) - (\text{tr} Z)^3)$.

### 3.1 Computing First-Order Derivatives

Higher-order derivatives of $c_k(Z)$ can be derived analogously. Here we try to find an efficient implementation of the computation of the first and second-order derivatives. It is possible to compute the first-order derivatives in $O(n^4)$ operations (multiplications and additions). Here, the main effort lies in the computation of $Z, Z^2, \ldots, Z^{n-k-1}$.

First, we compute the coefficients $c_k(Z)$ in the characteristic polynomial of $Z$. This can be done using the `poly` function in Matlab, but a more efficient algorithm due to Berkowitz [1] is faster and requires $O(n^4)$ operations. A simple Matlab implementation is given below

```matlab
function p = berkowitz (A)
    [n, m] = size (A);
    if n ~= m, error ('A must be square'); end;
    p = 1;
    for k = n: -1: 1,
        R = A(k,k+1:n);
        ci = [1 -A(k,k)];
        for i = k+1:n
            ci = [ci -R*A(k+1:n,k)];
            R = R*A(k+1:n,k+1:n);
        end
        p = toeplitz (ci, [1 zeros(1,n-k)]) * p;
    end
    end
```

Next, using the coefficients, $c_i(Z)$, we can compute the following series of matrices

$$C_k = c_n(Z) Z^{n-k-1} + c_{n-1}(Z) Z^{n-k-2} + \ldots + c_{k+1}(Z) I. \quad (9)$$
This is most efficiently computed iteratively as

\[ C_n = 0, \]
\[ C_{n-1} = I, \]
\[ C_{n-2} = Z + c_{n-1}(Z)I, \]
\[ \vdots \]
\[ C_k = C_{k+1}Z + c_{k+1}(Z)I, \]
\[ \vdots \]
\[ C_0 = C_1Z + c_1(Z)I, \]

which in total requires \( O(n^4) \) operations.

Now, the first-order derivatives of \( c_k(Z) \) can be computed as

\[ \frac{\partial c_k}{\partial x_i} = -\text{tr} C_k Z_i \]

where the computation of each coefficient requires \( O(n^2) \) operations. Consequently, \( n^2 \) derivatives can be computed in \( O(n^4) \) operations.

### 3.2 Computing Second-Order Derivatives

The second-order derivatives are given by

\[ \frac{\partial^2 c_k}{\partial x_i \partial x_j} = -\frac{\partial}{\partial x_j} \text{tr} C_k Z_i \]
\[ = -\text{tr} \frac{\partial C_k}{\partial x_j} Z_i - \text{tr} C_k Z_{ij} \]

The derivatives of \( C_i \) can be computed iteratively as

\[ \frac{\partial C_n}{\partial x_j} = 0, \]
\[ \frac{\partial C_{n-1}}{\partial x_j} = 0, \]
\[ \frac{\partial C_{n-2}}{\partial x_j} = Z_j + \frac{\partial c_{n-1}}{\partial x_j} I, \]
\[ \vdots \]
\[ \frac{\partial C_k}{\partial x_j} = \frac{\partial C_{k+1}}{\partial x_j} Z + C_{k+1}Z_j + \frac{\partial c_{k+1}}{\partial x_j} I, \]
\[ \vdots \]
\[ \frac{\partial C_0}{\partial x_j} = \frac{\partial C_1}{\partial x_j} Z + C_1Z_j + \frac{\partial c_1}{\partial x_j} I, \]
Consequently,
\[
\frac{\partial^2 c_k}{\partial x_i \partial x_j} = - \text{tr} \left( \frac{\partial C_{k+1}}{\partial x_j} Z + C_{k+1} Z_j + \frac{\partial C_{k+1}}{\partial x_j} I \right) Z_i - \text{tr} C_k Z_{ij}
\]
\[
= - \text{tr} \frac{\partial C_{k+1}}{\partial x_j} Z_i - \text{tr} C_{k+1} Z_j Z_i - \text{tr} C_k Z_{ij}
\]
\[
= - \text{tr} \left( \frac{\partial C_{k+2}}{\partial x_j} Z + C_{k+2} Z_j + \frac{\partial C_{k+2}}{\partial x_j} \right) Z Z_i
\]
\[
- \text{tr} C_{k+1} Z_j Z_i + \text{tr} C_k Z_{ij}
\]
\[
= - \text{tr} \frac{\partial C_{k+2}}{\partial x_j} Z^2 Z_i + (\text{tr} C_{k+2} Z_j) (\text{tr} Z Z_i) - \text{tr} C_{k+2} Z_j Z Z_i
\]
\[
- \text{tr} C_{k+1} Z_j Z_i + \text{tr} C_k Z_{ij} - \text{tr} C_k Z_{ik}.
\]

After reordering of terms we get,
\[
\frac{\partial^2 c_k}{\partial x_i \partial x_j} = (\text{tr} C_{n-1} Z_j) (\text{tr} Z^{n-k-2} Z_i) - \text{tr} C_{n-1} Z_j Z^{n-k-2} Z_i
\]
\[
+ \ldots
\]
\[
+ (\text{tr} C_{k+2} Z_j) (\text{tr} Z Z_i) - \text{tr} C_{k+2} Z_j Z Z_i
\]
\[
+ (\text{tr} C_{k+1} Z_j) (\text{tr} Z_i) - \text{tr} C_{k+1} Z_j Z_i
\]
\[
- \text{tr} C_k Z_{ik}.
\]

Without any structure in Z_i we need as much as O(n^6) operations to compute \( \frac{\partial^2 c_k}{\partial x_i \partial x_j} \), since the computation of all \( C_k Z_j \) and \( Z^k Z_j \) requires O(n^6) operations. This is based on the assumption that there are \( n^2 \) elements in \( Z \) such that \( i \) and \( j \) are from 1 to \( n^2 \).

### 3.3 Improving Computational Efficiency

With structure in \( Z \), we can reduce this. If \( Z_i = L_i R_i \) and \( Z_j = L_j R_j \) are assumed to contain single elements (ones). Here we assume that \( L \) and \( R \) are single-rank matrices. We can factor each terms as

\[
\text{tr} C_m Z_j Z^k Z_i = \text{tr} R_i C_m L_j R_j Z^k L_i = (\text{tr} R_i C_m L_j) (\text{tr} R_j Z^n L_i).
\]

Also,
\[
(\text{tr} C_m Z_j) (\text{tr} Z^k Z_i) = (\text{tr} R_j C_m L_j) (\text{tr} R_i Z^n L_i)
\]

Since we can compute \( C_m, C_{n-1}, \ldots, C_1 \) and \( Z, Z^2, \ldots, Z^{n-1} \) using \( 2(n-2) \) matrix \( n \times n \) multiplications we need \( O(n^3) \) operations. Using the structure of \( Z \), the trace operation, \( \text{tr} R_i C_m L_j \) can be replaced by a simple extraction of the appropriate element in \( C_m \) where \( R_i \) defines the row and \( L_j \) defines the column.

If \( Z \) is diagonal, we can reduce this to \( O(n^2) \) operations not including the operations needed to compute \( c_i \).

In the case when \( Z = I - XY \), the number of operations will be of the same order. Note that \( Z_i = -XY_i \), where one of \( X_i \) and \( Y_i \) is zero, and \( Z_{ij} = -X_i Y_j \).
For instance, the second-order derivatives of $c_{n-1}$ and $c_{n-2}$ become

$$\frac{\partial^2 c_{n-1}}{\partial x_i \partial x_j} = - \text{tr} Z_{ij},$$

and

$$\frac{\partial^2 c_{n-2}}{\partial x_i \partial x_j} = (\text{tr} Z_i)(\text{tr} Z_j) - \text{tr} Z_i Z_j - \text{tr} C_{n-2} Z_{ij}.$$

**References**
