A FAST ITERATIVE ORTHOGONALIZATION SCHEME FOR THE MACAULAY MATRIX *

KIM BATSELIER†, PHILIPPE DREESEN† AND BART DE MOOR †

Abstract. In this article we present a fast iterative orthogonalization scheme for two important subspaces of the Macaulay matrix: its range and left null space. It requires a graded monomial ordering and exploits the resulting structure of the Macaulay matrix induced by this graded ordering. The resulting orthogonal basis for the range will retain a similar structure as the Macaulay matrix and is as a consequence sparse. The computed orthogonal basis for the left null space is dense but typically has smaller dimensions. Two alternative implementations for the iterative orthogonalization scheme are presented: one using the singular value decomposition and another using a sparse rank revealing multifrontal QR decomposition. Numerical experiments show the effectiveness of the proposed iterative orthogonalization scheme in both running time and required memory compared to a standard orthogonalization. The sparse multifrontal QR implementation is superior in both total run time and required memory at the cost of being slightly less reliable for determining the numerical rank.

Key words. Macaulay matrix, orthogonal bases, range, null space, rank

AMS subject classifications. 15A03,15B05,15A18,15A23

1. Introduction. Many problems from algebraic geometry concerning multivariate polynomials can be phrased and solved in a numerical linear algebra setting. A few examples are: solving a system of multivariate polynomials [3, 12], multivariate polynomial division and elimination [4], computing an approximate LCM/GCD [2], computing a numerical Gröbner basis and solving the ideal membership problem [5]. The algorithms described in these cited articles have the general structure as shown in Algorithm 1.1. The goal is to compute some desired quantity X (e.g. the roots of the polynomial system or a polynomial from which certain variables are eliminated). The essential object in Algorithm 1.1 is the Macaulay matrix $M(d)$. This matrix, defined in Section 2, depends explicitly on the degree $d \in \mathbb{N}^+$ for which it is constructed. Note that we use a different convention in this article compared to the previously cited references. Coefficient vectors of polynomials will be column vectors in this article for the sake of readability as explained further on. The degree $d^\star$ for which X can be computed is in general not known beforehand. Algorithm 1.1 will therefore iterate over increasing values of the degree. In each iteration, the Macaulay matrix is constructed and orthogonal bases for either its range or left null space need to be computed. These orthogonal bases are then used to compute X. Since the Macaulay matrix is by definition rank-deficient, the first principal step is to determine its (numerical) rank. The most robust way to determine the rank and find the orthogonal

*Kim Batselier is a research assistant at the Katholieke Universiteit Leuven, Belgium. Philippe Dreesen is supported by the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen). Bart De Moor is a full professor at the Katholieke Universiteit Leuven, Belgium. Research supported by Research Council KUL: GOA/10/09 MaNet , PFV/10/002 (OPTEC), several PhD/postdoc & fellow grants, Flemish Government:IOF: IOF/KP/SCORES4CHEM,FWO: PhD/postdoc grants, projects: G.0588.09 (Brainmachine), G.0377.09 (Mechatronics MPC), G.0377.12 (Structured systems),IWT: PhD Grants, projects: SBO LeCoPro, SBO Climaqs, SBO POM, EUROSTARS SMART, iMinds 2012, Belgian Federal Science Policy Office: IUAP P7/19 (DYSCO, Dynamical systems, control and optimization, 2012-2017),EU: ERNSI, FP7-EMBOCON (ICT-248940), FP7-SADCO (MC ITN-264735), ERC ST HIGHWIND (259 166), ERC AdG A-DATADRIVE-B,COST: Action ICO806: IntelliCIS.

†Department of Electrical Engineering ESAT-SCD, KU Leuven / IBBT Future Health Department, 3001 Leuven, Belgium

bases is the singular value decomposition (SVD) [13]. A less computational expensive alternative is the rank-revealing QR decomposition. Both the SVD and the rank revealing QR decomposition are considered in this article. If for the current iteration the desired quantity X cannot be computed, then the degree d is incremented by one.

ALGORITHM 1.1. *General Algorithm*
***Input**: polynomial system $f_1, \ldots, f_s$ of degrees $d_1, \ldots, d_s$*
***Output**: desired output X*
  $X \leftarrow \emptyset$
  $d_0 \leftarrow \max(d_1, \ldots, d_s)$
  while** $X = \emptyset$ **do
    $M(i) \leftarrow$ *construct Macaulay matrix for degree i*
    $U \leftarrow$ *orthogonal basis for range of $M(i)$*
    $N \leftarrow$ *orthogonal basis for left null space of $M(i)$*
    $X \leftarrow$ *try to compute desired output from U and/or N*
    if** $X = \emptyset$ **then
      $d \leftarrow d + 1$
    end if
  end while

Previous implementations of this general algorithm do not exploit the structure of the Macaulay matrix nor do they use earlier computations when recomputing the orthogonal bases U, N for a higher degree. This is remedied by the iterative orthogonalization scheme presented in this article. The naive implementation of Algorithm 1.1 is considerably improved by

- reducing the computational complexity by working with a submatrix of $M(d)$ instead of the complete matrix,
- introducing an updating strategy for the orthogonal bases U, N which reuses the orthogonal bases from the previous iteration,
- introducing a sparse implementation which uses a sparse matrix data structure to reduce the required memory.

The outline of this article is as follows. In Section 2 a short overview of required definitions and notation is given. In Section 3 the main theorem and its proof are presented from which the orthogonalization scheme is derived. The computational complexity is analyzed in Section 4 for both Algorithm 1.1 and the proposed orthogonalization scheme. In Section 5 the application is discussed which will serve to illustrate the effectiveness of the proposed method from numerical experiments in Section 6. Finally, some concluding remarks are presented in Section 7.

2. Notation and Definitions. The vector space of all multivariate polynomials over n variables up to degree d over $\mathbb{C}$ is denoted by $\mathcal{C}_d^n$. Consequently the ring of all multivariate polynomials in n variables is denoted by $\mathcal{C}^n$. A canonical basis for this vector space consists of all monomials from degree 0 up to d. A monomial $x^a = x_1^{a_1} \ldots x_n^{a_n}$ has a multidegree $(a_1, \ldots, a_n) \in \mathbb{N}_0^n$ and (total) degree $|a| = \sum_{i=1}^n a_i$. The degree of a polynomial p then corresponds with the highest total degree of all monomials of p. Note that we can reconstruct the monomial x^a from its multidegree a. Furthermore, any ordering $>$ we establish on the space $\mathbb{N}_0^n$ will give us an ordering on monomials: if $a > b$ according to this ordering, we will also say that $x^a > x^b$. The orthogonalization scheme derived in Section 3 requires the use of a graded monomial ordering. Throughout the remainder of this article the follow-

ing graded ordering will be used. Note however that the iterative orthogonalization scheme works for any graded ordering.

DEFINITION 2.1. *Degree negative lexicographic. Let a and $b \in \mathbb{N}_0^n$. We say $a >_{dnlex} b$ if*

$$|a| = \sum_{i=1}^{n} a_i > |b| = \sum_{i=1}^{n} b_i, \text{ or } |a| = |b| \text{ and } a >_{nlex} b$$

where $a >_{nlex} b$ if, in the vector difference $a - b \in \mathbb{Z}^n$, the leftmost nonzero entry is negative.

The ordering is graded because it first compares the degrees of the two monomials and applies the negative lexicographic ordering when there is a tie. Once a monomial ordering $>$ is chosen, a polynomial can be uniquely identified with its coefficient vector as illustrated in the following example.

EXAMPLE 2.1. *The polynomial $f = 2 + 3x_1 - 4x_2 + x_1x_2 - 8x_1x_3 - 7x_2^2 + 3x_3^2$ in C_3^2 is represented by the vector*

$$f \;=\; \begin{matrix} 1 \\ x_1 \\ x_2 \\ x_3 \\ x_1^2 \\ x_1x_2 \\ x_1x_3 \\ x_2^2 \\ x_2x_3 \\ x_3^2 \end{matrix} \begin{pmatrix} 2 \\ 3 \\ -4 \\ 0 \\ 0 \\ 1 \\ -8 \\ -7 \\ 0 \\ 3 \end{pmatrix}$$

where the degree negative lex ordering of the monomials is indicated to the left of each coefficient.

By convention a coefficient vector will always be a column vector. Note that this is a different convention as used in [2, 4, 12]. This is for the sake of readability, it avoids the need to use the transpose of all matrices in the proof of the main theorem. Depending on the context we will use the label f for both a polynomial and its coefficient vector. $(.)^T$ will denote the transpose of the matrix or vector and I_a the square unit matrix of order a. Next we define the Macaulay matrix.

DEFINITION 2.2. *Given a set of polynomials $f_1, \ldots, f_s \in \mathcal{C}^n$, each of degree $d_i\,(i = 1, \ldots, s)$ then the Macaulay matrix of degree $d \geq \max(d_1, \ldots, d_s)$ is the matrix containing the coefficients of*

$$M(d) \;=\; \begin{pmatrix} f_1 & x_1 f_1 & \ldots & x_n^{d-d_1} f_1 & f_2 & x_1 f_2 & \ldots & x_n^{d-d_s} f_s \end{pmatrix} \tag{2.1}$$

where each polynomial f_i is multiplied with all monomials from degree 0 up to $d - d_i$ for all $i = 1, \ldots, s$.

We will use the convention that $M(d)$ is a $q \times p$ matrix whereas $M(d+1)$ is $q' \times p'$. This implies of course that $p' = p + \Delta\, p$ and $q' = q + \Delta\, q$. The dimension of the range of $M(d)$ is the rank r and the dimension of its left null space $m = q - r$. An orthogonal basis for the range of $M(d)$ is hence $q \times r$ and will be denoted by $U(d)$. Likewise, an orthogonal basis for the left null space of $M(d)$ is the $q \times m$ matrix $N(d)$. The Macaulay matrix is very sparse, especially for large degrees d. This sparsity is

quantified by the density of the matrix which is defined as the total number of nonzero entries of $M(d)$ divided by the total number of entries.

When a graded monomial ordering is used then there is always a column permutation P of the Macaulay matrix $M(d+1)$ such that

$$M(d+1)\,P \;=\; \begin{matrix} \\ q \\ \Delta q \end{matrix} \begin{matrix} \begin{matrix} p & \Delta p \end{matrix} \\ \begin{pmatrix} M(d) & M_a \\ 0 & M_b \end{pmatrix} \end{matrix}$$

where the M_b block contains all coefficients of monomials with total degree $d+1$. The graded monomial ordering hence results in a sparse block quasi-Toeplitz structured matrix. Now let

$$\begin{matrix} \\ m \\ \Delta q \end{matrix} \begin{matrix} \begin{matrix} \Delta p \end{matrix} \\ \begin{pmatrix} N(d)^T M_a \\ M_b \end{pmatrix} \end{matrix} \;=\; Q\,S\,V^T \tag{2.2}$$

be the SVD of $\left(M_a^T N(d)\; M_b^T\right)^T$. We denote the rank of this matrix by Δr. Suppose without loss of generality that $m + \Delta_q > \Delta_p$, then Q, S and V can be partitioned as

$$\begin{matrix} \\ m \\ \Delta q \end{matrix} \begin{matrix} \begin{matrix} \Delta p \end{matrix} \\ \begin{pmatrix} N(d)^T M_a \\ M_b \end{pmatrix} \end{matrix} \;=\; \begin{matrix} \\ m \\ \Delta q \end{matrix} \begin{matrix} \begin{matrix} \Delta r & m' \end{matrix} \\ \begin{pmatrix} L_1 & K_1 \\ L_2 & K_2 \end{pmatrix} \end{matrix} \begin{pmatrix} \Sigma & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix}$$

where Σ is diagonal and contains Δr nonzero singular values. This partitioning will be crucial in the proof of the main theorem in the next section.

3. The Orthogonalization Scheme. Now, all notation is in place to present the following main theorem. The iterative orthogonalization scheme of the Macaulay matrix is a direct application of this theorem.

THEOREM 3.1. *Let $U(d), N(d), M_a, M_b, L_1, L_2, K_1, K_2$ be the matrices as defined in Section 2. Then the following relationships hold:*

$$U(d+1) \;=\; \begin{pmatrix} U(d) & N(d)L_1 \\ 0 & L_2 \end{pmatrix} \quad \textit{and} \quad N(d+1) \;=\; \begin{pmatrix} N(d)K_1 \\ K_2 \end{pmatrix}.$$

Proof. For the sake of readability the degree (d) will be dropped from the notation $M(d), U(d), N(d)$. We start with the observation that the matrices

$$M(d+1) \;=\; \begin{pmatrix} M & M_a \\ 0 & M_b \end{pmatrix}, \quad \begin{pmatrix} U & M_a \\ 0 & M_b \end{pmatrix}$$

share the same range and left null space. Applying r Householder reflections onto the first r columns of the rightmost matrix results in

$$\begin{matrix} \\ r \\ m \\ \Delta q \end{matrix} \begin{matrix} \begin{matrix} q & \Delta q \end{matrix} \\ \begin{pmatrix} U^T & 0 \\ N^T & 0 \\ 0 & I_{\Delta q} \end{pmatrix} \end{matrix} \begin{matrix} \begin{matrix} r & \Delta p \end{matrix} \\ \begin{pmatrix} U & M_a \\ 0 & M_b \end{pmatrix} \end{matrix} \;=\; \begin{matrix} \begin{matrix} r & \Delta p \end{matrix} \\ \begin{pmatrix} I_r & U^T M_a \\ 0 & N^T M_a \\ 0 & M_b \end{pmatrix} \end{matrix}. \tag{3.1}$$

The orthogonal matrix can be moved to the right-hand side to obtain

$$\begin{matrix} \\ q \\ \Delta q \end{matrix} \begin{matrix} \begin{matrix} r & \Delta p \end{matrix} \\ \begin{pmatrix} U & M_a \\ 0 & M_b \end{pmatrix} \end{matrix} \;=\; \begin{matrix} \\ q \\ \Delta q \end{matrix} \begin{matrix} \begin{matrix} r & m & \Delta q \end{matrix} \\ \begin{pmatrix} U & N & 0 \\ 0 & 0 & I_{\Delta q} \end{pmatrix} \end{matrix} \begin{matrix} \begin{matrix} r & \Delta p \end{matrix} \\ \begin{pmatrix} I_r & U^T M_a \\ 0 & N^T M_a \\ 0 & M_b \end{pmatrix} \end{matrix}. \tag{3.2}$$

From (3.1) it is straightforward to see that the rank increase

$$\Delta r \triangleq \operatorname{rank} M(d+1) - \operatorname{rank} M(d))$$

is given by

$$\Delta r = \operatorname{rank} \begin{pmatrix} N^T M_a \\ M_b \end{pmatrix}$$

since

$$\begin{aligned} \operatorname{rank} M(d+1) &= \operatorname{rank} \begin{pmatrix} U & M_a \\ 0 & M_b \end{pmatrix} \\ &= \operatorname{rank} \begin{pmatrix} I_r & U^T M_a \\ 0 & N^T M_a \\ 0 & M_b \end{pmatrix}. \end{aligned}$$

The next step is to replace $\begin{pmatrix} M_a^T N & M_b^T \end{pmatrix}^T$ by its SVD in (3.2) to obtain

$$\left(\begin{array}{c|c} U & M_a \\ \hline 0 & M_b \end{array}\right) = \left(\begin{array}{c|cc} U & N & 0 \\ \hline 0 & 0 & I_{\Delta q} \end{array}\right) \left(\begin{array}{c|c} I_r & U^T M_a \\ \hline 0 & QSV^T \end{array}\right). \tag{3.3}$$

The Q can be factored out from the rightmost matrix in the following manner

$$\begin{pmatrix} I_r & U^T M_a \\ 0 & QSV^T \end{pmatrix} = \begin{array}{c} \\ r \\ m \\ \Delta q \end{array} \begin{array}{c} \begin{array}{ccc} r & \Delta r & m' \end{array} \\ \begin{pmatrix} I_r & 0 & 0 \\ 0 & L_1 & K_1 \\ 0 & L_2 & K_2 \end{pmatrix} \end{array} \begin{pmatrix} I_r & U^T M_a \\ 0 & \Sigma V_1^T \\ 0 & 0 \end{pmatrix}. \tag{3.4}$$

Note that since Δr is the increase in rank this implies that $m' = m + \Delta q - \Delta r = m + \Delta m$ is the dimension of the left null space of $M(d+1)$. Substituting (3.4) into (3.3) results in

$$\begin{pmatrix} U & M_a \\ 0 & M_b \end{pmatrix} = \begin{array}{c} \\ q \\ \Delta q \end{array} \begin{array}{c} \begin{array}{ccc} r & \Delta r & m' \end{array} \\ \begin{pmatrix} U & NL_1 & NK_1 \\ 0 & L_2 & K_2 \end{pmatrix} \end{array} \begin{pmatrix} I_r & U^T M_a \\ 0 & \Sigma V_1^T \\ 0 & 0 \end{pmatrix}. \tag{3.5}$$

The left matrix of the right-hand side is the product of two orthogonal matrices and hence also orthogonal. The theorem follows from (3.5). ∎

Observe that the orthogonal basis $U(d+1)$ retains a similar quasi-Toeplitz block structure as $M(d+1)$ and will therefore also be sparse. Theorem 3.1 can be immediately translated into Algorithm 3.1 to orthogonalize $M(d)$. The algorithm starts for the initial degree $d_0 \triangleq \max(d_1, \ldots, d_s)$. An orthogonal basis for the range and left null space of $M(d_0)$ are computed from its SVD. The subsequent steps of the algorithm are then to construct the extra rows $\begin{pmatrix} M_a^T & M_b^T \end{pmatrix}^T$ and update the orthogonal bases $U(d), N(d)$ using Theorem 3.1. Considering the sparsity of $M(d)$ and $U(d)$, it would be interesting to be able to use a sparse matrix data structure, like the column compressed form [9, p.8]. This avoids storing the large amount of zero entries in memory. A complete SVD is however not available for matrices using a sparse matrix data

structure but a rank revealing multifrontal QR decomposition [10] is. This allows us to replace the rank test of (2.2) by

$$\begin{pmatrix} N(d)^T M_a \\ M_b \end{pmatrix} = Q\, R\, P^T \tag{3.6}$$

where Q is orthogonal, R is upper triangular and P is a column permutation which reduces fill-in of R. Furthermore, if $\left(M_a^T\, N(d)\, M_b^T\right)^T$ is not of full column rank then R can be partitioned as

$$R = \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix}$$

such that the numerical rank can be estimated from the number of nonzero diagonal elements of R_{11}. This naturally leads to an implementation of Theorem 3.1 with matrices using a sparse matrix data structure. All SVDs are then replaced by sparse rank revealing multifrontal QR decompositions and (3.5) is then given by

$$\begin{pmatrix} U & M_a \\ 0 & M_b \end{pmatrix} = \begin{matrix} \\ q \\ \Delta q \end{matrix} \begin{matrix} \begin{matrix} r & \Delta\, r & m' \end{matrix} \\ \begin{pmatrix} U & NL_1 & NK_1 \\ 0 & L_2 & K_2 \end{pmatrix} \end{matrix} \begin{pmatrix} I_r & U^T M_a \\ 0 & R_{11}P_1^T + R_{12}P_2^T \\ 0 & 0 \end{pmatrix}.$$

ALGORITHM 3.1. *Iterative Orthogonalization of $M(d)$*
***Input**: polynomial system $f_1, \ldots, f_s$ of degrees $d_1, \ldots, d_s$, degree d*
***Output**: orthogonal bases $U(d), N(d)$*

$d_0 \leftarrow \max(d_1, \ldots, d_s)$
$U, N \leftarrow$ *orthogonal bases for range and left null space of* $M(d_0)$
for $i = d_0 + 1 \ldots d$ ***do***
  construct M_a, M_b *for degree* i
  $Q \leftarrow SVD(\left(M_a^T N\, M_b^T\right)^T)$ *or* $QR(\left(M_a^T N\, M_b^T\right)^T)$
  $\begin{pmatrix} L_1 & K_1 \\ L_2 & K_2 \end{pmatrix} \leftarrow Q$
  $U \leftarrow \begin{pmatrix} U & NL_1 \\ 0 & L_2 \end{pmatrix}$
  $N \leftarrow \begin{pmatrix} NK_1 \\ K_2 \end{pmatrix}$
end for

It is now possible to adjust Algorithm 1.1 such that it uses the iterative orthogonalization scheme of Theorem 3.1. The pseudo-code for this update is shown in Algorithm 3.2.

ALGORITHM 3.2. *Updated Algorithm Macaulay matrix*
***Input**: polynomial system $f_1, \ldots, f_s$ of degrees $d_1, \ldots, d_s$*
***Output**: desired output X*

```
d ← max(d_1, ..., d_s)
U, N ← orthogonal bases for range and left null space of of M(d)
X ← try to compute desired output from U and/or N
if X = ∅ then
  d ← d + 1
end if
while X = ∅ do
  construct M_a, M_b for degree d
  Q ← SVD( (M_a^T N M_b^T)^T ) or QR( (M_a^T N M_b^T)^T )
  U, N ← update U, N using Q and Theorem 3.1
  X ← try to compute desired output from U and N
  if X = ∅ then
    d ← d + 1
  end if
end while
```

4. Computational Complexity. The most expensive computational step in both Algorithm 1.1 and Algorithm 3.2 is the computation of the orthogonal bases. In this section an estimate of the gain in computational complexity is derived for both the SVD and sparse QR-based implementation.

4.1. SVD. We first provide an estimate on the total number of operations for computing the SVD of a complete $M(d)$. The number of rows q and the number of columns p of $M(d)$ are the following polynomials in d :

$$q(d) = \binom{d+n}{n} = \frac{d^n}{n!} + O(d^{n-1}),$$
$$p(q) = \sum_{i=1}^{s} \binom{d-d_i+n}{n} = \frac{s\,d^n}{n!} + O(d^{n-1}).$$

From these expressions it is easily seen that for most degrees the number of rows will be smaller than the number of columns. We will assume from here on that the number of columns is always bigger than the number of rows. In each iteration of Algorithm 1.1 the SVD is computed from the complete Macaulay matrix. Not the full SVD is required however. Only the left singular vectors Q and the diagonal matrix S are needed. This takes about $4p(d)q(d)^2 + 8q(d)^3$ operations [13, p. 254]. We approximate both $q(d), p(d)$ by their highest order term in order to have an estimate on their order of magnitude. Substituting this into the expression for the number of operations results in the following estimate for the computational cost of the SVD in Algorithm 1.1

$$4\left(\frac{sd^n}{n!}\right)\left(\frac{d^n}{n!}\right)^2 + 8\left(\frac{d^n}{n!}\right)^3 = \frac{4(s+2)}{(n!)^3}\,d^{3n}. \tag{4.1}$$

The SVD-step in Algorithm 3.2 is applied on a $(m + \Delta q) \times \Delta p$ matrix. $m(d)$ is a polynomial of maximal degree $n - 1$. The reason for this is explained in algebraic

geometry where $m(d)$ is called the projective Hilbert Polynomial [8, p.462]. The degree of this polynomial is the dimension of the solution set of $f_1, \ldots, f_s$, which is maximally $n-1$. Obviously, the highest order terms of $\Delta p(d)$ and $\Delta q(d)$ are $sd^{n-1}/(n-1!)$ and $d^{n-1}/(n-1!)$ respectively. Retaining the highest order terms in the expression for the total amount of operations gives us the following estimate for the cost of the SVD in Algorithm 3.2

$$4\left(\frac{sd^{n-1}}{(n-1)!}\right)\left(\frac{d^{n-1}}{(n-1)!}\right)^2 + 8\left(\frac{d^{n-1}}{(n-1)!}\right)^3 = \frac{4(s+2)}{(n-1)!^3}\,d^{3n-3}. \tag{4.2}$$

Dividing (4.1) by (4.2) leads to an estimated gain of d^3/n^3 operations when using the iterative orthogonalization scheme. This gain can be quite substantial when the degree d is large. Memory is still the bottleneck for orthogonalizing $M(d)$ for large degrees however. Although the SVD of a smaller submatrix needs to be computed, it still grows with $\approx O(d^{n-1})$. This polynomial growth is unfortunately inherent to problems which involve multivariate polynomials.

4.2. rank revealing QR decomposition. In order to describe the computational complexity of the rank revealing QR decomposition the assumption is made that Businger-Golub column pivoting [7] is used. This serves as an upper bound on the complexity since it does not take the sparsity pattern of the Macaulay matrix into account. In practice, a sparse multifrontal QR decomposition algorithm can be used which will exploit the structure.

For a $q(d) \times p(d)$ Macaulay matrix of rank $r(d)$, the computational complexity of the Businger-Golub QR factorization is given by $4p(d)q(d)r(d) - 2r(d)^2(p(d)+q(d)) + 4r(d)^3/3$ [13, p. 250]. In this expression $r(d)$ can be replaced by $q(d) - m(d)$ where $m(d)$ is again a polynomial of maximal degree $n-1$. Like before, only higher order terms are retained for $q(d), p(d)$ and substituted into $4p(d)q(d)r(d) - 2r(d)^2(p(d) + q(d)) + 4r(d)^3/3$ to obtain

$$4\,\frac{sd^{3n}}{(n!)^3} - 2(s+1)\frac{d^{3n}}{(n!)^3} + \frac{4}{3}\,\frac{d^{3n}}{(n!)^3} = \frac{2(3s-1)}{3(n!)^3}\,d^{3n}. \tag{4.3}$$

The same reasoning can be applied for the $(m + \Delta q) \times \Delta p$ submatrix which leads to

$$4\,\frac{sd^{3n-3}}{((n-1)!)^3} - 2(s+1)\frac{d^{3n-3}}{((n-1)!)^3} + \frac{4}{3}\,\frac{d^{3n-3}}{((n-1)!)^3} = \frac{2(3s-1)}{3((n-1)!)^3}\,d^{3n-3}$$

and the same gain of d^3/n^3 operations when using the iterative orthogonalization scheme. Comparing (4.1) with (4.3) reveals that the QR decomposition will be about

$$\frac{6(s+2)}{(3s-1)}$$

times faster than the SVD. This factor reaches 3 for $s = 5$ and then asymptotically approaches 2 in the limit for large s.

5. Application. In Section 6 the performance for Algorithm 1.1 and its updated version, Algorithm 3.2, will be compared for the following application.

PROBLEM 5.1. *Given a multivariate polynomial system $f_1, \ldots, f_s \in \mathcal{C}^n$ with a finite number of affine roots and a monomial x_i ($i \in \{1, \ldots, s\}$). Find a univariate polynomial $f(x_i)$ which lies in the polynomial ideal $\langle f_1, \ldots, f_s \rangle$.*

The requirement that the polynomial system $f_1, \ldots, f_s$ has a finite number of affine roots is essential since otherwise it is not guaranteed that for every monomial x_i such a univariate polynomial $f(x_i)$ exists. As described in [4], $f(x_i)$ lies in the intersection

$$\text{range}(U(d)) \cap \text{span}\{1, x_i, x_i^2, x_i^3, \ldots, x_i^d\}$$

for an unknown degree d. This requirement is easily understood:

$$f(x_i) \in \text{range}(U(d))$$

since it belongs to the polynomial ideal $\langle f_1, \ldots, f_s \rangle$ and

$$f(x_i) \in \text{span}\{1, x_i, x_i^2, x_i^3, \ldots, x_i^d\}$$

implies that it is univariate in x_i. The coefficient matrix of a canonical basis for $\text{span}\{1, x_i, x_i^2, x_i^3, \ldots, x_i^d\}$ will be denoted by $E(d)$. Note that this matrix is orthogonal. Algorithm 1.1 and 3.2 can then be used to find $X = f(x_i)$. Every iteration one needs to check whether there is a non-empty intersection between the range of $M(d)$ and $E(d)$. This can be done by inspecting the smallest principal angle between these two vector spaces. When this angle is zero, $f(x_i)$ can then be computed as a basis vector for the intersection. As described in [6, p. 582-583] and [14, p. 6], the sine of the smallest principal angle between the range of $U(d)$ and $E(d)$ is the the smallest singular value μ_{d+1} of $E(d) - U(d)\,U(d)^T\,E(d)$. The columns of this $q(d) \times d + 1$ matrix span the orthogonal projection of $\text{span}\{1, x_i, x_i^2, x_i^3, \ldots, x_i^d\}$ onto the orthogonal complement of range$(U(d))$. If there is a non-empty intersection, then an orthonormal basis vector for this vector space is the right singular vector z_{d+1} corresponding with μ_{d+1}. This basis vector is hence also the desired univariate polynomial $f(x_i)$. Observe that $E(d) - U(d)\,U(d)^T\,E(d)$ can be replaced by $N(d)\,N(d)^T E(d)$ since

$$\begin{aligned} E(d) - U(d)\,U(d)^T\,E(d) &= (I - U(d)\,U(d)^T)\,E(d) \\ &= N(d)\,N(d)^T\,E(d) \end{aligned}$$

which means that $U(d)$ never needs to be explicitly kept during the iterations. The right singular vectors for $N(d)\,N(d)^T E(d)$ are identical to the right singular vectors of the $m(d) \times (d+1)$ matrix $N(d)^T E(d)$ which is much smaller than the original $q(d) \times (d+1)$ matrix. The complete algorithm to solve Problem 5.1 using the iterative orthogonalization scheme is presented in Algorithm 5.1. A tolerance τ is introduced which is used to decide the numerical rank and whether the principal angle is numerically zero. Note that some steps like the initial SVD/QR of $M(d)$ and the construction of the extra columns M_a, M_b are not mentioned anymore for the sake of readability. They are assumed implicitly. Algorithm 5.1 to solve Problem 5.1 is easily derived using the structure of Algorithm 1.1.

ALGORITHM 5.1. *Solving Problem 5.1*
***Input**: polynomials $f_1, \ldots, f_s$ of degrees $d_1, \ldots, d_s$, monomial x_i, tolerance τ*
***Output**: univariate $f(x_i)$*

```
f(x_i) ← ∅
d ← max(d_1, ..., d_s)
N ← orthogonal bases for left null space of M(d)
E ← canonical basis for span{1, x_i, x_i^2, x_i^3, ..., x_i^d}
[W S Z] ← SVD(N(d)^T E(d))
if arcsin(μ_{d+1}) < τ then
   f(x_i) ← z_{d+1}
else
   d ← d + 1
end if
while f(x_i) = ∅ do
   N ← update N using Theorem 3.1
   E ← canonical basis for span{1, x_i, x_i^2, x_i^3, ..., x_i^d}
   [W S Z] ← SVD(N(d)^T E(d))
   if arcsin(μ_{d+1}) < τ then
      f(x_i) ← z_{d+1}
   else
      d ← d + 1
   end if
end while
```

5.1. Choosing the numerical tolerance τ. We assume that all computations are performed in double precision. This sets the machine precision to $\epsilon \approx 2.22 \times 10^{-16}$. A crucial step in the iterative algorithm is the determination of the numerical rank. The determination of an incorrect numerical rank during one of the iterations affects all consequent iterations. A good choice for the numerical tolerance is therefore of the utmost importance to guarantee a correct result.

For the SVD-based approach, numerical experiments indicate that a 'standard' choice of $\tau = \max(q(d), p(d))\, ||M(d)||_2\, \epsilon$ works fairly well for most cases. Let $\sigma_1 \geq \ldots \geq \sigma_{m+\Delta p}$ be the singular values of $\left(M_a^T N\, M_b^T\right)^T$, then the numerical rank Δr is chosen such that

$$\sigma_1 \geq \ldots \geq \sigma_{\Delta r} \geq \tau \geq \sigma_{\Delta r+1} \geq \ldots \geq \sigma_{m+\Delta p}.$$

The approx-rank gap $\sigma_{\Delta r}/\sigma_{\Delta r+1}$ [15, p.920] then serves as a measure of how well the numerical rank is defined. Indeed, if there is a large gap between $\sigma_{\Delta r}$ and $\sigma_{\Delta r+1}$ and τ lies between these two values then small changes in τ will not affect the determination of the numerical rank. When the default value for the numerical tolerance fails, one could try to determine the numerical rank such that the approxi-rank gap is maximal. This will be explored in numerical experiments in Section 6.

The sparse multifrontal QR decomposition method from [10], SuiteSparseQR, uses the numerical tolerance $\tau = 20\,(m + \Delta p + \Delta q)\,\epsilon\, D$ where D is the largest 2-norm of any row of $\left(M_a^T N\, M_b^T\right)^T$. The numerical rank is then estimated as the number of nonzero diagonal entries. The rank-revealing QR decomposition is known to be less reliable for the rank determination. It is reported in [10] that SuiteSparseQR is able to correctly determine the correct numerical rank for about two-thirds of the rank deficient matrices in the University of Florida Sparse Matrix Collection [11]. The

numerical rank for many of those matrices is ill-defined. If the numerical rank was very well-defined (approxi-rank gap $> 10^3$), then 95 % of the time the numerical rank was correctly determined.

We have observed that dense polynomial systems for which every possible monomial has a nonzero coefficient tend to produce Macaulay matrices for which it is difficult to determine the rank. The difficulty lies not in a small approxi-rank gap but rather in a failing of the default choices for the tolerance. For these cases it is recommended for the user to manually check either the singular values or the diagonal elements of R. This is illustrated in the next section.

6. Numerical Experiments. The orthogonalization scheme of Algorithm 3.1 has been implemented in Matlab [16] and is freely available on request. All numerical experiments were performed on a 2.66 GHz quad-core desktop computer with 8 GB RAM using 64 bit Matlab. The effectiveness of the orthogonalization scheme is illustrated by comparing the run times of Algorithms 1.1 (using the SVD) and 5.1 when solving Problem 5.1 for both the SVD and sparse QR-based approach. Let $\tilde{f}(x_i)$ be the numerical result of either Algorithm 1.1 or 5.1 and $f(x_i)$ the exact result. Then the forward error $e \triangleq ||\tilde{f}(x_i) - f(x_i)||_2$ will serve as a measure for the numerical accuracy of the two algorithms. The exact result $f(x_i)$ was computed in Maple(TM) [1]. Since $\tilde{f}(x_i)$ is a unit vector, $f(x_i)$ is first divided by its 2-norm before computing e. In addition to the run times, the required memory to store the matrices during the orthogonalization is reported for $d = d^\star$. For Algorithm 1.1 this is the memory required to store $M(d^\star)$. Algorithm 3.1 needs to store $U(d^\star - 1), N(d^\star - 1), \begin{pmatrix} M_a^T & M_b^T \end{pmatrix}^T$ using either a dense or sparse matrix data structure. The total amount of required memory to store these three matrices will also be reported for the full and sparse case. Note that for the elimination algorithm, as mentioned before, it is actually not necessary to keep the $U(d)$ matrices. The reported memory requirement in this section however is for the full orthogonalization of $M(d)$.

6.1. Example 1. In the first numerical experiment the capability of the algorithms to deal with high total degrees is tested. The polynomial system consists of 3 polynomials in 3 unknowns of total degree 12

$$\left\{ \begin{array}{lrcl} f_1: & x_1^{12} + x_2^{12} + x_3^{12} - 4 & = & 0 \\ f_2: & x_1^{12} + 2\,x_2^{12} - 5 & = & 0 \\ f_3: & x_1^6\,x_3^6 - 1 & = & 0. \end{array} \right.$$

The corresponding univariate polynomials are all of degree 24 and are found at $d^\star = 24$. The coefficient vectors of f_1, f_2, f_3 are very sparse and hence the Macaulay matrix is also very sparse. The density of the 2925×1365 matrix $M(24)$ is 0.13%. The sparse multifrontal QR method is therefore expected to perform better which is indeed seen from Table 6.1. The sparse multifrontal QR method runs respectively 15 and 5 times faster compared to Algorithm 1.1 and the SVD-based Algorithm 5.1. The forward errors e are all below the numerical tolerance $\tau \approx 10^{-13}$(SVD), $\tau \approx 10^{-12}$(QR). Table 6.2 reports the required memory in order to find the orthogonal bases U, N for $d = 24$. Surprisingly, the SVD-based iterative orthogonalization needs more memory than computing the SVD of $M(24)$. In this particular case the orthogonal bases $U(23), N(23)$ and $\begin{pmatrix} M_a^T & M_b^T \end{pmatrix}^T$ are very sparse. This explains why using a sparse matrix data structure for the same matrices requires almost 300 times less memory.

TABLE 6.1
Run Times and Errors: Example 1

	Alg. 1.1 [seconds]	Alg. 5.1 SVD [seconds]	Alg. 5.1 QR [seconds]	e Alg 1.1	e SVD	e QR
$f(x_1)$	15.23	5.10	1.05	2.49e−16	7.56e−16	1.35e−15
$f(x_2)$	14.84	4.79	1.00	8.71e−16	2.40e−16	5.98e−16
$f(x_3)$	14.89	4.81	1.00	1.04e−15	4.66e−16	3.31e−15

TABLE 6.2
Required memory for inputs of orthogonalization: Example 1

$d^\star$	$M(d^\star)$ [MB]	Alg. 3.1 full [MB]	Alg. 3.1 sparse [MB]
24	30.46	57.67	0.1966

6.2. Example 2. For the next numerical experiment the number of variables is increased to 6 and the degrees are maximally 4:

$$\begin{cases} x_1^2 + x_3^2 - 1 &= 0 \\ x_2^2 + x_4^2 - 1 &= 0 \\ x_5\, x_3^3 + x_6\, x_4^3 - 1.2 &= 0 \\ x_5\, x_1^3 + x_6\, x_2^3 - 1.2 &= 0 \\ x_5\, x_3^2\, x_1 + x_6\, x_4^2\, x_2 - 0.7 &= 0 \\ x_5\, x_3\, x_1^2 + x_6\, x_4\, x_2^2 - 0.7 &= 0. \end{cases}$$

The univariate polynomials $f(x_1)$ up to $f(x_4)$ are of degree 4 and $f(x_5), f(x_6)$ are of degree 2. They are all found for $d^\star = 10$ at which $M(10)$ is 9702×8008. Again, the polynomials of the given system are very sparse resulting in a density of $M(10)$ of 0.037%. From Table 6.3 it is seen that the sparse multifrontal QR method is again the fasted with the same numerical accuracy. The numerical tolerance for the SVD-based approach is $\tau \approx 10^{-12}$ and for the QR-based approach $\tau \approx 10^{-11}$. Note that only $f(x_1)$ and $f(x_5)$ are reported since the results are the same for the remaining variables. The iterative QR method is approximately 36 and 4.5 times faster than Algorithm 1.1 and the iterative SVD method respectively. The gain in required memory when using the sparse matrix data structure compared to the dense naive and dense iterative orthogonalization is about 13 and 6 times respectively.

TABLE 6.3
Run Times and Errors: Example 2

	Alg. 1.1 [seconds]	Alg. 5.1 SVD [seconds]	Alg. 5.1 QR [seconds]	e Alg 1.1	e SVD	e QR
$f(x_1)$	1035.28	138.29	28.61	1.91e−14	1.09e−14	4.29e−14
$f(x_5)$	1059.94	136.83	29.40	4.41e−14	2.31e−13	6.12e−15

TABLE 6.4
Required memory for inputs of orthogonalization: Example 2

$d^\star$	$M(d^\star)$ [MB]	Alg. 3.1 full [MB]	Alg. 3.1 sparse [MB]
10	592.75	290.09	44.21

6.3. Example 3. The number of variables and polynomials is further increased to 10 polynomials in 10 variables of total degree 2

$$\begin{cases} 5\,x_1\,x_2 + 5\,x_1 + 3\,x_2 + 55 = 0 \\ 7\,x_2\,x_3 + 9\,x_2 + 9\,x_3 + 19 = 0 \\ 3\,x_3\,x_4 + 6\,x_3 + 5\,x_4 - 4 = 0 \\ 6\,x_4\,x_5 + 6\,x_4 + 7\,x_5 + 118 = 0 \\ x_5\,x_6 + 3\,x_5 + 9\,x_6 + 27 = 0 \\ 6\,x_6\,x_7 + 7\,x_6 + x_7 + 72 = 0 \\ 9\,x_7\,x_8 + 7\,x_7 + x_8 + 35 = 0 \\ 4\,x_8\,x_9 + 4\,x_8 + 6\,x_9 + 16 = 0 \\ 8\,x_9\,x_{10} + 4\,x_9 + 3\,x_{10} - 51 = 0 \\ 3\,x_1\,x_{10} - 6\,x_1 + x_{10} + 5 = 0. \end{cases}$$

The corresponding univariate polynomials are also all of second degree and are found at $d^\star = 6$. The Macaulay matrix $M(6)$ is then 8008×10010 with a density of 0.05%. Again, the run times and forward errors for only $f(x_1), f(x_3), f(x_5)$ are reported in Table 6.5 for the same reason as in Example 6.2. The iterative sparse QR-based method is respectively about 33 and 5 times faster than Algorithm 1.1 and the iterative SVD-based method and requires about 10 times less memory. Numerical tolerances are $\tau \approx 10^{-12}$ for the SVD-based method and $\tau \approx 10^{-11}$ for the QR-based method. The forward errors are of the same order of magnitude or smaller than these tolerances.

TABLE 6.5
Run Times and Errors: Example 3

	Alg. 1.1 [seconds]	Alg. 5.1 SVD [seconds]	Alg. 5.1 QR [seconds]	e Alg 1.1	e SVD	e QR
$f(x_1)$	2234.77	372.26	66.92	4.85e−13	3.24e−13	1.92e−13
$f(x_3)$	2218.01	365.34	65.77	2.76e−12	5.84e−12	3.92e−12
$f(x_5)$	2234.98	365.51	66.07	4.30e−13	2.33e−13	8.40e−14

TABLE 6.6
Required memory for inputs of orthogonalization: Example 3

$d^\star$	$M(d^\star)$ [MB]	Alg. 3.1 full [MB]	Alg. 3.1 sparse [MB]
6	611.57	505.64	50.32

6.4. Example 4. In this example we illustrate the failure of the default numerical tolerances when working with dense polynomial systems. The polynomial system

consists of 3 polynomials, each of degree 4. Each polynomial consists of 35 terms, corresponding to all possible monomials in 3 variables from degrees 0 up to 4. The integer coefficients are uniformly drawn from the interval $[-100, 100]$. Normalizing the coefficient vector of each polynomial such that it is a unit vector keeps the singular values small. For this particular instance the rang test fails at degree $d = 23$. The matrix $\left(M_a^T N\, M_b^T\right)^T$ is then 570×340. We first discuss the SVD-based algorithm. The numerical tolerance is $\tau = 1.26 \times 10^{-13}$. The singular values σ_{276} up to σ_{280} are

$$\{0.1266, 1.72 \times 10^{-13}, 1.30 \times 10^{-13}, 1.05 \times 10^{-13}, 5.01 \times 10^{-14}\}.$$

Using the default tolerance would determine the numerical rank to be 278 with a corresponding approxi-rank gap of 1.24. The numerical rank however is well-determined to be 276 with an approxi-rank gap of $7.34\,10^{11}$. It is clear from this example that maximizing the approxi-rank gap over all pairs of consecutive singular values would correctly retrieve the numerical rank. Care needs to be taken on how this maximization is carried out since it is possible that there is more than one large gap.

The QR-based algorithm suffers from the same problem. In contrast to the singular values, the diagonal entries of R, denoted by r_{ii}, are not all positive nor sorted in descending order. We therefore denote the values of $|r_{ii}|$, sorted in descending order, by s_j where j runs from 1 to $\min(m + \Delta q, \Delta p)$. One can then do a similar analysis on s_j as with the singular values. At $d = 10$ the default numerical tolerance is $\tau = 1.11 \times 10^{-12}$ and the last two nonzero elements of s_j for $d = 10$ are $s_{78} = 0.044, s_{79} = 2.79 \times 10^{-12}$. The numerical rank determined from using the default tolerance is 79 while the singular values indicate it should be 78 with an approxi-rank gap of 2.34×10^{12}. Maximizing the ratio of two consecutive nonzero s_j values would also in this case retrieve the correct numerical rank.

6.5. Example 5. The following polynomial system demonstrates the failure of the rank revealing QR decomposition to correctly determine the rank. As will be shown for the degree $d = 8$, this failure is not related to the default values of the numerical tolerances. Consider the following polynomial system in $\mathcal{C}_2^4$:

$$\begin{cases} 2\,x_4^2 + 2\,x_3^2 + 2\,y^2 + 2\,x_1^2 - x_1 &= 0 \\ 2\,x_3\,x_4 + 2\,x_2\,x_3 + 2\,x_1\,x_2 - x_2 &= 0 \\ 2\,x_2\,x_4 + 2\,x_1\,x_3 + x_2^2 - x_3 &= 0 \\ 2\,x_4 + 2\,x_3 + 2\,x_2 + x_1 - 1 &= 0. \end{cases}$$

At $d = 7$ the default tolerance for the QR-based orthogonalization fails. Inspecting $s_{120} =$ and $s_{121} =$ shows that the numerical rank should be 120 instead of 121 although the default tolerance is $\tau = 1.69 \times 10^{-12}$. The SVD-based method confirms the numerical rank of 120 with an approxi-rank gap of $\sigma_{120}/\sigma_{121} = 2.85 \times 10^{14}$. At $d = 8$ the numerical rank is estimated by the rank revealing QR to be 166. Indeed, the numerical tolerance is $\tau = 2.27 \times 10^{-12}$ and $s_{166} =$. The SVD-based algorithm however shows that the numerical rank is 165 and well determined with an approxi-rank gap $\sigma_{165}/\sigma_{166} = 2.10 \times 10^{14}$.

Remarkably, in spite of this wrong numerical rank, the QR-based elimination algorithm still retrieves the correct univariate polynomials as demonstrated in Table 6.7. Both $f(x_1)$ and $f(x_4)$ are of degree 8 and are found at $d = 8$. The numerical tolerances for $d = 8$ are $\approx 10^{-12}$ and all forward errors are a few orders of magnitude larger. This indicates a loss of accuracy due to numerical computations. This loss of accuracy is largest for the sparse QR-based algorithm. Due to the relatively small

size of $\left(M_a^T N\ M_b^T\right)^T$, 252×8, there is only a modest improvement in both total run time and required memory (Table 6.8) when using the iterative orthogonalization.

TABLE 6.7
Run Times and Errors: Example 5

	Alg. 1.1 [seconds]	Alg. 5.1 SVD [seconds]	Alg. 5.1 QR [seconds]	e Alg 1.1	e SVD	e QR
$f(x_1)$	2.81	1.75	1.54	1.22e−10	1.37e−10	7.05e−8
$f(x_4)$	2.97	1.61	1.69	7.46e−11	1.35e−10	1.90e−8

TABLE 6.8
Required memory for inputs of orthogonalization: Example 5

$d^\star$	$M(d^\star)$ [MB]	Alg. 3.1 full [MB]	Alg. 3.1 sparse [MB]
8	3.62	2.23	1.01

7. Conclusion. We have presented an iterative algorithm which computes an orthogonal bases for the range and the left null space of the Macaulay matrix. Both an SVD-based and sparse QR-based implementation were discussed. Numerical experiments indicate that the sparse QR-based implementation runs up to 30 times faster and uses at least 10 times less memory compared to a naive full orthogonalization using Algorithm 1.1. This significant gain in run time and required memory comes at the cost of a slightly less reliable rank test compared to the SVD as demonstrated in Example 5. The performance of the SVD-based implementation lies in between the performance of Algorithm 1.1 and the QR-based implementation.

8. Acknowledgements.

REFERENCES

[1] MAPLE 16, *Maplesoft, a division of Waterloo Maple Inc*, 2012. Waterloo, Ontario.
[2] K. BATSELIER, P. DREESEN, AND B. DE MOOR, *A geometrical approach to finding multivariate approximate LCMs and GCDs*. Accepted for publication in Linear Algebra and its Applications, 2012.
[3] K. BATSELIER, P. DREESEN, AND B. DE MOOR, *Prediction Error Method Identification is an Eigenvalue Problem*, Proc 16th IFAC Symposium on System Identification (SYSID 2012), 2012, pp. 221–226.
[4] K. BATSELIER, P. DREESEN, AND B. DE MOOR, *The Geometry of Multivariate Polynomial Division and Elimination*. Accepted for publication in SIAM Journal on Matrix Analysis and Applications, 2012.
[5] K BATSELIER, P DREESEN, AND B DE MOOR, *Numerical Polynomial Algebra: The Canonical Decomposition and Numerical Gröbner Bases*. Submitted to SIMAX, 2013.
[6] Å BJÖRCK AND G. H. GOLUB, *Numerical Methods for Computing Angles Between Linear Subspaces*, Mathematics of Computation, 27 (1973), pp. pp. 579–594.
[7] P. BUSINGER AND G.H. GOLUB, *Linear Least Squares Solutions by Householder Transformations*, Numerische Mathematik, 7 (1965), pp. 269–276.
[8] D. A. COX, J. B. LITTLE, AND D. O'SHEA, *Ideals, Varieties and Algorithms*, Springer-Verlag, third ed., 2007.
[9] T. A. DAVIS, *Direct Methods for Sparse Linear Systems (Fundamentals of Algorithms 2)*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2006.

[10] ———, *Algorithm 915, SuiteSparseQR: Multifrontal Multithreaded Rank-Revealing Sparse QR Factorization*, ACM Transactions on Mathematical Software, 38 (2011).
[11] T. A. Davis and Y. Hu, *The University of Florida Sparse Matrix Collection*, ACM Trans. Math. Softw., 38 (2011), pp. 1:1–1:25.
[12] P. Dreesen, K. Batselier, and B. De Moor, *Back to the roots: Polynomial system solving, linear algebra, systems theory*, Proc 16th IFAC Symposium on System Identification (SYSID 2012), 2012, pp. 1203–1208.
[13] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, 3rd ed., Oct. 1996.
[14] A.V. Knyazev and M.E. Argentati, *Principal angles between subspaces in an A-based scalar product: Algorithms and perturbation estimates*, SIAM Journal on Scientific Computing, 23 (2002), pp. 2008–2040.
[15] T.Y. Li and Z. Zeng, *A rank-revealing method with updating, downdating, and applications*, SIAM Journal on Matrix Analysis and Applications, 26 (2005), pp. 918–946.
[16] MATLAB R2012a, *The Mathworks Inc.*, 2012. Natick, Massachusetts.