MATHEMATICAL ENGINEERING TECHNICAL REPORTS

Index Reduction for Differential-Algebraic Equations with Mixed Matrices

Satoru IWATA, Taihei OKI and Mizuyo TAKAMATSU

METR 2017–23 December 2017

Index Reduction for Differential-Algebraic Equations with Mixed Matrices[*]

Satoru Iwata[†] Taihei Oki[†] Mizuyo Takamatsu[‡]

Abstract

Differential-algebraic equations (DAEs) are widely used for modeling of dynamical systems. The difficulty in numerically solving a DAE is measured by its differentiation index. For highly accurate simulation of dynamical systems, it is important to convert high index DAEs into low index DAEs. Most of existing simulation software packages for dynamical systems are equipped with an index reduction algorithm given by Mattsson and Söderlind. Unfortunately, this algorithm fails if there are unlucky numerical cancellations.

These numerical cancellations are often caused by accurate numbers in structural equations. Distinguishing those accurate numbers from inaccurate ones that represent physical characteristics, Murota and Iri introduced the notion of a mixed matrix as a mathematical tool for faithful model description in structural approach to systems analysis. For DAEs described with the use of mixed matrices, efficient algorithms to compute the index have been developed by exploiting matroid theory.

This paper presents an index reduction algorithm for linear DAEs whose coefficient matrices are mixed matrices. Our algorithm detects numerical cancellations between accurate constants, and transforms a DAE into an equivalent DAE to which Mattsson–Söderlind's index reduction algorithm is applicable. The algorithm is based on the combinatorial relaxation approach, which is a framework to solve a linear algebraic problem by iteratively relaxing it into an efficiently solvable combinatorial optimization problem. The algorithm relies not on symbolic manipulations but on fast combinatorial algorithms on graphs and matroids. Our algorithm is proved to work for any linear DAEs whose coefficient matrices are mixed matrices, and is expected to preserve the sparsity of DAEs. We further provide an improved algorithm under an assumption based on dimensional analysis of dynamical systems. Our algorithms can be applied to nonlinear DAEs by regarding nonlinear terms as inaccurate numbers.

1 Introduction

An l-th order *differential-algebraic equation* (DAE) for $x\colon \mathbb{R} \to \mathbb{R}^n$ is a differential equation in the form of

$$F\bigl(t, x(t), \dot{x}(t), \ldots, x^{(l)}(t)\bigr) = 0, \tag{1}$$

where $F\colon \mathbb{R} \times \mathbb{R}^n \times \cdots \times \mathbb{R}^n \to \mathbb{R}^n$ is a sufficiently smooth function. DAEs have aspects of both ordinary differential equations (ODEs) $\dot{x}(t) = \varphi(t, x(t))$ and algebraic equations $G(t, x(t)) = 0$. DAEs are widely used for modeling of dynamical systems such as mechanical systems, electrical circuits and chemical reaction plants.

A DAE (1) can be rewritten as a first order DAE

$$F(t, x(t), \dot{x}(t)) = 0 \tag{2}$$

by replacing higher order derivatives of x with newly introduced variables. The difficulty in numerically solving the DAE (2) is measured by its *differentiation index* [1], which is defined as the minimum nonnegative integer ν such that the system of equations

$$F(t, x(t), \dot{x}(t)) = 0, \quad \frac{\mathrm{d}}{\mathrm{d}t}F(t, x(t), \dot{x}(t)) = 0, \quad \ldots, \quad \frac{\mathrm{d}^\nu}{\mathrm{d}t^\nu}F(t, x(t), \dot{x}(t)) = 0$$

[*]This work was supported in part by JST CREST, Grant Number JPMJCR14D2, Japan.

[†]Department of Mathematical Informatics, Graduate School of Information Science and Technology, The University of Tokyo, Hongo 7-3-1, Bunkyo-ku, Tokyo 113-8656, Japan. E-mail: {iwata, taihei_oki}@mist.i.u-tokyo.ac.jp

[‡]Department of Information and System Engineering, Chuo University, Kasuga 1-13-27, Bunkyo-ku, Tokyo 112-8551, Japan. E-mail: takamatsu@ise.chuo-u.ac.jp

can determine $\dot{x}$ as a continuous function of t and x. In other words, ν is the number of times one has to differentiate the DAE (2) to get an ODE. Intuitively, the differentiation index represents how far the DAE is from ODEs.

A common approach for solving a high (≥ 2) index DAE is to convert it into a low (≤ 1) index DAE. This process is called an *index reduction*, which is important for accurate simulation of dynamical systems. Most of existing simulation software packages for dynamical systems, such as Dymola, OpenModelica, MapleSim and Simulink, are equipped with the index reduction algorithm given by Mattsson–Söderlind [10] (MS-algorithm). The MS-algorithm uses Pantelides' method [21] as preprocessing. Pantelides' method constructs a bipartite graph from structural information of a given DAE, and solves an assignment problem on the bipartite graph efficiently. The MS-algorithm then differentiates equations in the DAE with the aid of the information obtained by Pantelides' method, and replaces some derivatives with dummy variables. The MS-algorithm returns a sparse DAE if a given DAE is sparse, and thus the algorithm can be applied to large scale DAEs.

Pantelides' method, however, does not work even for the following simple DAE

$$\begin{cases} \dot{x}_1 + \dot{x}_2 + x_3 = 0, \\ \dot{x}_1 + \dot{x}_2 \phantom{{}+x_3} = 0, \\ \phantom{\dot{x}_1 + {}} x_2 + \dot{x}_3 = 0. \end{cases}$$

Pantelides' algorithm reports that the index is zero whereas it is indeed two. This is because the method cannot detect the singularity of the coefficient matrix $\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ of $\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix}$. As this toy example shows, Pantelides' method, which ignores numerical information, may fail on some DAEs due to numerical cancellations. This kind of failure can also occur in other methods to reduce the index or to analyze DAEs such as a structural algorithm of Unger et al. [25] and the Σ-method of Pryce [22].

Some index reduction algorithms address this problem. One example is the $\sigma\nu$-method by Chowdhry et al. [2], which is based on the algorithm by Unger et al. [25]. The method performs the Gaussian elimination on the Jacobian matrix $\partial F / \partial \dot{x}$ under the assumption that nonlinear or time-varying terms do not cancel out. Tan et al. [24] presented two methods called LC-method and ES-method, which improve the Σ-method of Pryce. Their methods identify numeric or symbolic cancellations and modify the DAE if necessary.

Pantelides' method [21] and the Σ-method [22] discard numerical information, which sometimes leads to a failure of the methods. In dynamical systems, specific numbers in structural equations such as the conservation laws should be treated numerically, while we can deal with physical characteristic values as nonzero parameters without reference to their nominal values. For a faithful model of a dynamical system, it is natural to distinguish accurate and inaccurate numbers. This led Murota–Iri [18] to introduce the notion of a *mixed matrix*, which is a matrix consisting of the following two kinds of entries:

Accurate Constants, which represent precise values such as coefficients of conservation laws. We assume that arithmetic operations of these constants can be performed in constant time.

Independent Parameters, which are algebraically independent over the field of accurate constants. These parameters often represent physical quantities such as masses, lengths or electric resistances since their nominal values are inaccurate by measurement noises and other errors. These parameters should be treated combinatorially without reference to their nominal values.

For example, consider an electric network consisting of voltage sources, resistances and wires connecting them. A linear equation representing the circuit has two kinds of coefficients: the exact '± 1's coming from Kirchhoff's law, and the resistance values coming from Ohm's law. Since the nominal values of resistances are supposed to be inaccurate, it is natural to model the system by a linear equation with a mixed matrix, where constants and parameters represent the exact '± 1's and the resistances, respectively. See an example in Section 6.2 for modeling of an RLC circuit with a mixed matrix.

Mathematical aspects of mixed matrices are studied by Murota [13]. If all nonzero entries of a matrix are independent parameters, then its rank is equal to the maximum size of a matching in an associated bipartite graph. For a mixed matrix, the rank computation corresponds to solving an independent matching problem on matroids, which is a generalization of the maximum matching problem on bipartite graphs. An efficient algorithm based on matroid theory is provided for the rank computation of mixed matrices; see [17] for detail.

In this paper, we provide an index reduction algorithm for a linear DAE

$$\sum_{k=0}^{l} A_k x^{(k)}(t) = f(t) \tag{3}$$

with $n \times n$ mixed matrices $A_0, A_1, \ldots, A_l$ and a sufficiently smooth function $f\colon \mathbb{R} \to \mathbb{R}^n$.

Modeling dynamical systems by mixed matrices is advantageous for the index reduction as follows. First, we can resolve numerical cancellations in a mixed matrix by modifying only accurate constants since the cancellations do not occur involving independent parameters. Thus our algorithm is expected to preserve the sparsity of the matrix compared to other index reduction algorithms that transform the entire matrix. Next, since accurate constants arising from typical dynamical systems are integers or rational numbers, we can avoid arithmetic operations and comparisons of floating-point numbers. This fact makes the algorithm stable. Finally, if we want to simulate a dynamical system on many different settings of physical quantities, we can reuse the resulting low-index DAE as long as the values of physical quantities do not unluckily cancel out.

Our approach for the index reduction is to transform a DAE into an equivalent DAE to which the MS-algorithm is applicable, based on the *combinatorial relaxation* method, introduced by Murota [12, 15] as a framework to solve a linear algebraic problem by iteratively relaxing it into an efficiently solvable combinatorial optimization problem. Our algorithm does not rely on symbolic manipulations, while it uses numerical calculations and fast combinatorial optimization algorithms on graphs and matroids. For an l-th order DAE, our algorithm is proved to run in $\mathrm{O}\big(l^2 n^{\omega+2}\big)$ time, where ω is the matrix multiplication exponent, i.e., the number of arithmetical operations needed to multiply two $n \times n$ matrices is $\mathrm{O}(n^\omega)$. The current best known value of ω is almost 2.3728639 due to [9]. In practice, however, one can adopt $\omega = 3$. Our algorithm is expected to run much faster in most cases because the algorithm terminates without modifying a DAE unless it has unlucky numerical cancellations.

In addition, we give an improved algorithm for DAEs whose coefficients are *dimensionally consistent*. The dimensional consistency, which is introduced by Murota [11], is a mathematical assumption on mixed matrices reflecting the principle of dimensional homogeneity in physical systems, and DAEs arising from dynamical systems naturally ensure this assumption. We show that the improved algorithm retains the dimensional consistency, and that the running time is $\mathrm{O}\big(ln^4 \log n\big)$. Since the order l of typical DAEs is not so large, we claim that the algorithm does not significantly increase the computational cost compared to the MS-algorithm, which costs $\mathrm{O}\big(n^4 + ln^2\big)$ time by the straightforward matrix multiplication.

Furthermore, though our index reduction algorithm is designed for linear DAEs, the algorithm can be applied to nonlinear DAEs by regarding nonlinear terms as independent parameters. While the similar approach is adopted in $\sigma\nu$-method of Chowdhry et al. [2], our method is expected to be applicable to a larger class of nonlinear DAEs than the $\sigma\nu$-method because our method does not transform DAEs involving nonlinear terms.

We describe the relation between the proposed algorithm and related index reduction algorithms. If all nonzero entries of $A(s)$ are independent parameters, our algorithm just passes a given DAE to the MS-method as we described above. In contrast, if $A(s)$ has no independent parameter, then our algorithm coincides with the LC-method by Tan et al. [24]. In addition, our algorithm is similar to the $\sigma\nu$-method [2] in the sense that both methods treat matrices having accurate constants and independent parameters, yet their approaches are quite different; the $\sigma\nu$-method is based on the Gaussian elimination approach by Gear [5], whereas our algorithm relies on the dummy variable approach by Mattsson–Söderlind [10].

A recent work [6] has proposed an index reduction algorithm which is proved to work for any instances of first order linear DAEs with constant coefficients. The algorithm directly reduces the index of a given DAE by row operations, whereas our algorithm only resolves numerical cancellations in a DAE and eventually relies on the MS-algorithm for the actual index reduction process. Thus our algorithm is expected to preserve the sparsity of DAEs compared to the algorithm in [6].

This paper is organized as follows. Section 2 reviews the previous index computation and reduction algorithms for linear DAEs with constant coefficients, including the MS-algorithm and combinatorial relaxation algorithms. Section 3 explains mixed matrices and their rank identities. Section 4 describes the proposed algorithm. Section 5 improves our algorithm under the assumption of the dimensional consistency. Section 6 illustrates two examples. Section 7 discusses an application to nonlinear DAEs. Finally, Section 8 concludes this paper.

2 Index Reduction for Linear DAEs

2.1 Index of Linear DAEs

A linear DAE with constant coefficients is the following DAE:

$$\sum_{k=0}^{l} A_k x^{(k)}(t) = f(t), \tag{4}$$

where $A_0, A_1, \dots, A_l$ are $n \times n$ matrices and $f\colon \mathbb{R} \to \mathbb{R}^n$ is a sufficiently smooth function. We assume that f is Laplace transformable for simplicity, though this assumption is not essential. By the Laplace transformation, the DAE (4) is transformed into

$$A(s)\tilde{x}(s) = \tilde{f}(s) + \sum_{k=0}^{l}\sum_{i=1}^{k} s^{k-i} A_k x^{(i-1)}(0), \tag{5}$$

where $\tilde{x}(s)$ and $\tilde{f}(s)$ are the Laplace transforms of $x(t)$ and $f(t)$, respectively, and $A(s) = \sum_{k=0}^{l} s^k A_k$. We henceforth denote the right-hand side of (5) by $\hat{f}(s)$. The matrix $A(s)$ is a matrix whose entries are polynomials, called a *polynomial matrix*. We say that $A(s)$ is *nonsingular* if its determinant is not identically zero.

An initial value $\left(x_0, x_0^{(1)}, \dots, x_0^{(l-1)}\right) \in \mathbb{R}^n \times \cdots \times \mathbb{R}^n$ is said to be *consistent* if there exists at least one solution of (4) satisfying

$$x(0) = x_0,\ \dot{x}(0) = x_0^{(1)},\ \dots,\ x^{(l-1)}(0) = x_0^{(l-1)}. \tag{6}$$

We say that the DAE (4) is *solvable* if there exists a unique solution of (4) satisfying the initial value condition (6) for an arbitrary consistent point. The solvability of (4) is characterized by $A(s)$ as follows.

Theorem 2.1 ([1, 23]). A linear DAE (4) is solvable if and only if the associated polynomial matrix $A(s)$ is nonsingular.

See [1, Theorem 2.3.1] for $l = 1$ and [23, Theorems 2.22–23] for $l \geq 2$. In this paper, we focus on solvable DAEs (4). In addition, we abuse the term "DAE" and also refer to the equation (5) as a DAE.

The differentiation index of a first order linear DAE (4) with $A(s) = A_0 + sA_1$ is known to equal

$$\nu(A) = \delta_{n-1}(A) - \delta_n(A) + 1. \tag{7}$$

Here, $\delta_k(A)$ denotes the maximum degree of the determinant of a submatrix in $A(s)$ of size k, i.e.,

$$\delta_k(A) = \max\{\deg\det A(s)[I, J] \mid |I| = |J| = k\},$$

where $A(s)[I, J]$ is the submatrix in $A(s)$ with row set I and column set J, and $\deg p(s)$ designates the degree of a polynomial $p(s)$ in s. In particular, $\delta_n(A)$ is the degree of the determinant of $A(s)$, and $\delta_{n-1}(A)$ is the maximum degree of a cofactor of $A(s)$. For a DAE (4) with $l \geq 2$, its index is defined to be that of a first order DAE obtained by replacing higher order derivatives with new variables [24].

2.2 Assignment Problem

In analysis of DAEs, Pryce [22] introduced an assignment problem as a reinterpretation of Pantelides' algorithm [21] as follows.

Consider a linear DAE (5) with $n \times n$ nonsingular polynomial matrix $A(s)$ with row set R and column set C. We denote the (i, j) entry of $A(s)$ by $A_{i,j}(s)$. Let $G(A)$ denote the bipartite graph with vertex set $R \cup C$ and edge set $E(A) = \{(i, j) \in R \times C \mid A_{i,j}(s) \neq 0\}$. An edge subset $M \subseteq E(A)$ is called a *matching* if the ends of edges in M are disjoint. Since $A(s)$ is nonsingular, $G(A)$ has a matching of size n, called a *perfect matching*. We set the weight c_e of an edge $e = (i, j) \in E(A)$ by $c_e = c_{i,j} = \deg A_{i,j}(s)$.

The assignment problem on $G(A)$ is the following problem $\mathrm{P}(A)$:

$$\mathrm{P}(A)\quad \left|\ \begin{array}{ll} \text{maximize} & \displaystyle\sum_{e \in M} c_e \\ \text{subject to} & M \subseteq E(A) \text{ is a perfect matching on } G(A). \end{array}\right.$$

The dual problem $\mathrm{D}(A)$ of $\mathrm{P}(A)$ is expressed as follows:

$$\mathrm{D}(A)\quad\left|\begin{array}{lll}\text{minimize} & \displaystyle\sum_{j\in C} q_j - \sum_{i\in R} p_i & \\ \text{subject to} & q_j - p_i \geq c_{i,j} & ((i,j)\in E(A)), \\ & p_i \in \mathbb{Z} & (i\in R), \\ & q_j \in \mathbb{Z} & (j \in C).\end{array}\right.$$

We denote the optimal value of the problem $\mathrm{P}(A)$ (and $\mathrm{D}(A)$) by $\hat{\delta}_n(A)$. Recall that $\delta_n(A)$ denotes $\deg\det A(s)$. It is well known that $\delta_n(A) \leq \hat{\delta}_n(A)$ holds, and the equality is attained if and only if the coefficient of $s^{\hat{\delta}_n(A)}$ in $\det A(s)$ does not vanish. In this sense, $\hat{\delta}_n(A)$ serves as a combinatorial upper bound on $\delta_n(A)$. We call $A(s)$ *upper-tight* if $\delta_n(A) = \hat{\delta}_n(A)$ holds.

For a dual feasible solution (p,q), a *tight coefficient matrix* $A^{\#}$ of $A(s)$ is defined by

$$A^{\#}_{i,j} \coloneqq \text{the coefficient of } s^{q_j - p_i} \text{ in } A_{i,j}(s)$$

for each $i \in R$ and $j \in C$. Note that $A^{\#}$ changes depending on (p,q). This matrix is called a "Σ-Jacobian" by Pryce [22]; the name "tight coefficient matrix" is due to Murota [14].

2.3 Computing the Index via Combinatorial Relaxation

The tight coefficient matrix plays an important role in the combinatorial relaxation algorithm of Murota [15] to compute $\delta_n(A)$ for a polynomial matrix $A(s)$ through the following lemma.

Lemma 2.2 ([12, Proposition 6.2])**.** Let $A(s)$ be an $n \times n$ nonsingular polynomial matrix and $A^{\#}$ the tight coefficient matrix of $A(s)$ with respect to an optimal solution of $\mathrm{D}(A)$. Then $A(s)$ is upper-tight if and only if $A^{\#}$ is nonsingular.

The combinatorial relaxation method consists of the following three phases.

Phase 1. Compute a combinatorial upper bound $\hat{\delta}_n(A)$ of $\delta_n(A)$ by solving an assignment problem.

Phase 2. Check whether $A(s)$ is upper-tight using Lemma 2.2. If it is, return the estimation and halt.

Phase 3. Modify $A(s)$ to improve $\hat{\delta}_n(A)$ by replacing $A(s)$ with $U(s)A(s)$, where $U(s)$ is a *unimodular matrix.* Go back to Phase 2.

Here, a unimodular matrix is a square polynomial matrix whose determinant is a nonzero constant. The algorithm is designed so that $\hat{\delta}_n(A)$ decreases in each iteration, while unimodular transformations preserve $\delta_n(A)$. Thus, after a finite number of iterations, the algorithm terminates with $\hat{\delta}_n(A) = \delta_n(A)$.

Subsequently, Murota [14] applied the combinatorial relaxation approach to computing $\delta_k(A)$ for $k = 1, \ldots, n$. In this algorithm, Phase 3 modifies $A(s)$ to $U(s)A(s)V(s)$, where $U(s)$ and $V(s)$ are *biproper Laurent polynomial matrices*, i.e., entries are all polynomials in $1/s$ and the determinants are nonzero constants. This type of transformation is known to preserve $\delta_k(A)$. The values of $\delta_{n-1}(A)$ and $\delta_n(A)$ determine the index $\nu(A)$ by (7).

2.4 Mattsson–Söderlind's Index Reduction Algorithm

We now review Mattsson–Söderlind's index reduction algorithm applied to a linear DAE (5) with $n \times n$ nonsingular polynomial matrix $A(s)$. Let (p,q) be an optimal solution of $\mathrm{D}(A)$. For $h \in \mathbb{Z}$, we define

$$R_h \coloneqq \{i \in R \mid p_i = h\}, \quad R_{\geq h} \coloneqq \{i \in R \mid p_i \geq h\},$$
$$C_h \coloneqq \{j \in C \mid q_j = h\}, \quad C_{\geq h} \coloneqq \{j \in C \mid q_j \geq h\}.$$

The MS-algorithm applied to the DAE (5) is outlined as follows. The following description is a rewritten version especially for linear DAEs, though the original MS-algorithm is designed for nonlinear DAEs [10, Section 3.1].

Mattsson–Söderlind's Index Reduction Algorithm

Step 1. Compute an optimal solution (p,q) of $\mathrm{D}(A)$ satisfying $p_i, q_j \geq 0$ for $i \in R$ and $j \in C$. Let $A^{\#}$ denote the tight coefficient matrix of $A(s)$ with respect to (p,q). If $A^{\#}$ is singular, then the algorithm terminates in failure.

Step 2. For each $h = 0, \dots, \eta + 1$ $\bigl(\eta := \max_{i \in R} p_i\bigr)$, obtain $J_h \subseteq C_{\geq h}$ such that $A^{\#}[R_{\geq h}, J_h]$ is nonsingular and

$$C = J_0 \supseteq J_1 \supseteq J_2 \supseteq \cdots \supseteq J_\eta \supseteq J_{\eta+1} = \varnothing.$$

Step 3. For each $j \in C$, let k_j be an integer such that $j \in J_{k_j}$ and $j \notin J_{k_j+1}$. Introduce k_j dummy variables $z_j^{[q_j]}, z_j^{[q_j-1]}, \dots, z_j^{[q_j-k_j+1]}$ corresponding to $s^{q_j}\tilde{x}_j$, $s^{q_j-1}\tilde{x}_j, \dots, s^{q_j-k_j+1}\tilde{x}_j$, respectively.

Step 4. For each $i \in R$, return the 0-th, 1-st, ..., p_i-th order derivatives of the i-th equation. Replace variables with the corresponding dummy variables.

The validity of the MS-algorithm is established as follows.

Proposition 2.3 ([10, Section 3.2])**.** Let $A(s)$ be a polynomial matrix in the DAE (5) and $A^{\#}$ the tight coefficient matrix of $A(s)$ with respect to an optimal solution of $\mathrm{D}(A)$. If $A^{\#}$ is nonsingular, then the MS-algorithm correctly returns an equivalent DAE with index at most one.

From Lemma 2.2, the condition in Proposition 2.3 is equivalent to the upper-tightness of $A(s)$ as follows.

Corollary 2.4. Let $A(s)$ be a polynomial matrix in the DAE (5). If $A(s)$ is upper-tight, then the MS-algorithm correctly returns an equivalent DAE with index at most one.

The description above is still valid for a nonlinear DAE (1) by redefining $A(s)$ as

$$\text{the coefficient of } s^k \text{ in } A_{i,j}(s) := \text{the partial derivative of the } i\text{-th equation with respect to } x_j^{(k)}$$

for each $i = 1, \dots, n$, $j = 1, \dots, n$ and $k = 0, \dots, l$. Then the nonsingularity of $A^{\#}$ essentially comes from the requirement of the implicit function theorem, which is used to convert the DAE into an ODE by solving the DAE for the highest order derivatives.

2.5 Index Reduction via Combinatorial Relaxation

For a linear DAE (5) that does not satisfy the validity condition of the MS-algorithm, we need to modify it to apply the MS-algorithm. Here, the modification of DAEs must preserve the sets of their solutions. We can use unimodular transformations in the form of

$$U(s)A(s)\tilde{x}(s) = U(s)\hat{f}(s),$$

where $U(s)$ is a unimodular matrix. Since unimodular transformations correspond to the operations of adding an equation in the DAE or its (higher order) derivative to another equation, the DAEs before and after the transformation have the same solution set.

Murota's combinatorial relaxation algorithm [15] for computing $\delta_n(A)$ described in Section 2.3 modifies a given polynomial matrix $A(s)$ into an upper-tight polynomial matrix $\bar{A}(s) = U(s)A(s)$ using some unimodular matrix $U(s)$. Then from Corollary 2.4, the matrix $\bar{A}(s)$ satisfies the validity condition of the MS-algorithm. Therefore, we can use Murota's algorithm as an index reduction algorithm by combining with the MS-algorithm. Note that this modification may change $\delta_{n-1}(A)$, and hence $\nu(A)$. We also remark that this index reduction algorithm indeed coincides with the LC-method of Tan et al. [24] applied to the linear DAEs with constant coefficients.

3 DAEs with Mixed Matrices

The algorithms explained in Section 2 work under the assumption that one can exactly perform arithmetic operations of inaccurate physical quantities, which naturally arise from dynamical systems. In order to distinguish inaccurate quantities from accurate numbers, we deal with a DAE with mixed matrices.

3.1 Mixed Matrices and Mixed Polynomial Matrices

Let $\mathbf{F}$ be a field and $\mathbf{K}$ a subfield of $\mathbf{F}$. A typical setting in the context of DAEs is $\mathbf{F} = \mathbb{R}$ and $\mathbf{K} = \mathbb{Q}$. A matrix T over $\mathbf{F}$ is said to be *generic* if the set of nonzero entries of T is algebraically independent over $\mathbf{K}$. A *mixed matrix* with respect to $(\mathbf{K}, \mathbf{F})$ is a matrix in the form of $Q + T$, where Q is a matrix over $\mathbf{K}$ and T is a generic matrix. A *layered mixed matrix* (or *LM-matrix*) is a mixed matrix such that nonzero rows of Q and T are disjoint. An LM-matrix A can be expressed as $A = \binom{Q}{T}$.

A polynomial matrix $A(s) = \sum_{k=0}^{l} s^k A_k$ is called a *mixed polynomial matrix* if it is expressed as $A_k = Q_k + T_k$ with Q_k and T_k that satisfy the following conditions:

(MP-Q) Each Q_k $(k = 0, \ldots, l)$ is a matrix over $\mathbf{K}$.

(MP-T) The set of nonzero entries of $T_0, \ldots, T_l$ is algebraically independent over $\mathbf{K}$.

A *layered mixed polynomial matrix* (or *LM-polynomial matrix*) is a mixed polynomial matrix such that nonzero rows of $Q(s) = \sum_{k=0}^{l} s^k Q_k$ and $T(s) = \sum_{k=0}^{l} s^k T_k$ are disjoint. An LM-polynomial matrix is expressed as $A(s) = \binom{Q(s)}{T(s)}$.

3.2 Rank of LM-matrices

For a matrix A, we denote the row set and column set by $\mathrm{Row}(A)$ and $\mathrm{Col}(A)$, respectively. Consider the associated bipartite graph $G(A) = (R, C; E(A))$, where $R = \mathrm{Row}(A)$ and $C = \mathrm{Col}(A)$. The *term-rank* of A is the maximum size of a matching in $G(A)$, and is denoted by $\text{t-rank}\, A$. It is well known that $\mathrm{rank}\, A \le \text{t-rank}\, A$ holds. The equality is attained if and only if A has a submatrix of size $\text{t-rank}\, A$ with nonzero determinant. This is analogous to the relation between δ_n and $\hat{\delta}_n$ for a polynomial matrix.

Let $A = \binom{Q}{T}$ be an LM-matrix. If A has no accurate constants, i.e., A is a generic matrix T, it holds that $\mathrm{rank}\, T = \text{t-rank}\, T$ from the independence of nonzero entries. From this equality, we can compute $\mathrm{rank}\, T$ by solving a maximum matching problem on the associated bipartite graph $G(T)$. For general LM-matrices, the following holds from the generalized Laplace expansion.

Proposition 3.1 ([19, Theorem 3.1]). For an LM-matrix $A = \binom{Q}{T}$ with $R_Q = \mathrm{Row}(Q)$, $R_T = \mathrm{Row}(T)$ and $C = \mathrm{Col}(A)$, the following rank identity holds:

$$\mathrm{rank}\, A = \max\{\mathrm{rank}\, Q[R_Q, J] + \text{t-rank}\, T[R_T, C \setminus J] \mid J \subseteq C\}. \tag{8}$$

The problem of maximizing the right-hand side of (8) can be reduced to an *independent matching problem* on a matroid; see [17, Section 4.2] for detail. The following identity is obtained from the duality of the independent matching problem.

Proposition 3.2 ([19, Theorem 3.1]). For an LM-matrix $A = \binom{Q}{T}$ with $R_Q = \mathrm{Row}(Q)$, $R_T = \mathrm{Row}(T)$ and $C = \mathrm{Col}(A)$, the following rank identity holds:

$$\mathrm{rank}\, A = \min\{\mathrm{rank}\, Q[R_Q, J] + \text{t-rank}\, T[R_T, J] + |C \setminus J| \mid J \subseteq C\}. \tag{9}$$

Similarly, we give the following term-rank identity for LM-matrices, which will be used later in the proof of Lemma 4.7.

Proposition 3.3. For an LM-matrix $A = \binom{Q}{T}$ with $R_Q = \mathrm{Row}(Q)$, $R_T = \mathrm{Row}(T)$ and $C = \mathrm{Col}(A)$, the following term-rank identity holds:

$$\text{t-rank}\, A = \min\{\text{t-rank}\, Q[R_Q, J] + \text{t-rank}\, T[R_T, J] + |C \setminus J| \mid J \subseteq C\}.$$

Proof. This immediately follows from the well-known rank formula of a union matroid [4] and the fact that the union of transversal matroids is also a transversal matroid [20, Corollary 11.3.8]. □

3.3 Combinatorial Relaxation Algorithm for Mixed Polynomial Matrices

Murota [16] described the first algorithm to compute $\delta_k(A)$ for a mixed polynomial matrix $A(s)$ through a reduction to a *valuated independent assignment problem.* A combinatorial relaxation algorithm for the computation of $\delta_k(A)$ is given in [7]. This algorithm first converts a mixed polynomial matrix into an LM-polynomial matrix $A(s) = \binom{Q(s)}{T(s)}$, and modifies $A(s)$ to

$$\bar{A}(s) = \begin{pmatrix} U_Q(s) & O \\ O & I \end{pmatrix} \begin{pmatrix} Q(s) \\ T(s) \end{pmatrix},$$

where I is an identity matrix of appropriate size and $U_Q(s)$ is a *Laurent polynomial matrix.* With the use of (7), we can obtain the index $\nu(A)$ by computing $\delta_n(A)$ and $\delta_{n-1}(A)$.

In order to devise an index reduction algorithm for DAEs with mixed matrices, we need to make use of unimodular transformations instead of Laurent polynomial transformations, as explained in Section 2.5.

4 Combinatorial Relaxation Algorithm for Index Reduction with Mixed Polynomial Matrices

This section presents our index reduction algorithm for a DAE

$$A(s)\tilde{x}(s) = \hat{f}(s) \tag{10}$$

with a nonsingular mixed polynomial matrix $A(s)$, which is the Laplace transform of the DAE (3). From Corollary 2.4, our goal is to find a unimodular matrix $U(s)$ such that $\bar{A}(s) = U(s)A(s)$ is upper-tight. Then applying the MS-algorithm to the DAE $U(s)A(s)\tilde{x}(s) = U(s)\hat{f}(s)$, we obtain a resultant low-index DAE.

We cannot perform row operations on $A(s)$ involving rows containing independent parameters. Our first step is to convert a given DAE (10) into another DAE whose coefficient matrix $A(s)$ is an LM-polynomial matrix expressed as $A(s) = \binom{Q(s)}{T(s)}$. Then we can transform $A(s)$ to

$$\bar{A}(s) = \begin{pmatrix} U_Q(s) & O \\ O & I \end{pmatrix} \begin{pmatrix} Q(s) \\ T(s) \end{pmatrix}, \tag{11}$$

where $U_Q(s)$ is a unimodular matrix. Note that we are allowed to perform row operations only on $Q(s)$ even for an LM-polynomial matrix $A(s) = \binom{Q(s)}{T(s)}$, and thus we cannot always reduce the index to one only by row operations on $Q(s)$. We describe this conversion process from mixed polynomial matrices into LM-polynomial matrices in Section 4.1.

After the conversion, we find a unimodular matrix $U_Q(s)$ in (11) such that $A(s)$ is upper-tight based on the combinatorial relaxation approach. The outline of our algorithm is shown as follows.

Algorithm for Tightness

Phase 1. Construct an optimal solution (p, q) of $\mathrm{D}(A)$.

Phase 2. If the tight coefficient matrix $A^{\#}$ with respect to (p, q) is nonsingular, then return $A(s)$ and halt.

Phase 3. Modify $A(s)$ into $\bar{A}(s)$ such that $\hat{\delta}_n(\bar{A}) \leq \hat{\delta}_n(A) - 1$ and $\delta_n(A) = \delta_n(\bar{A})$. Update (p, q) to an optimal solution of $\mathrm{D}(\bar{A})$, and go back to Phase 2.

Section 4.2 describes an algorithm to find (p, q) in Phase 1. The condition in Phase 2, which is equivalent to the upper-tightness of $A(s)$ by Lemma 2.2, can be checked by solving an independent matching problem [19]. The matrix modification and an update procedure of (p, q) in Phase 3 is explained in Sections 4.3 and 4.4, respectively. In Section 4.5, we analyze the time complexity of our algorithm.

4.1 Reduction to LM-polynomial Matrices

We first convert the DAE (10) with a mixed polynomial coefficient matrix $A(s) = Q(s) + T(s)$ into the following augmented DAE

$$\begin{pmatrix} I & Q(s) \\ -D & DT(s) \end{pmatrix} \begin{pmatrix} \tilde{y}(s) \\ \tilde{z}(s) \end{pmatrix} = \begin{pmatrix} \hat{f}(s) \\ 0 \end{pmatrix}, \tag{12}$$

where D is a diagonal matrix whose diagonal entries are independent parameters $\tau_1, \ldots, \tau_n$. Note that the coefficient matrix of the augmented DAE (12) is an LM-polynomial matrix as the set of nonzero coefficients of entries in $-D$ and $DT(s)$ is algebraically independent over $\mathbf{K}$.

Proposition 4.1. Let $\binom{\tilde{y}(s)}{\tilde{z}(s)}$ be a solution of the DAE (12). Then $\tilde{z}(s)$ is a solution of the DAE (10).

Proof. By left-multiplying both sides of (12) by a nonsingular constant matrix $\begin{pmatrix} I & O \\ I & D^{-1} \end{pmatrix}$, we obtain

$$\begin{pmatrix} I & Q(s) \\ O & A(s) \end{pmatrix} \begin{pmatrix} \tilde{y}(s) \\ \tilde{z}(s) \end{pmatrix} = \begin{pmatrix} \hat{f}(s) \\ \hat{f}(s) \end{pmatrix},$$

where O is a zero matrix. Thus it holds $A(s)\tilde{z}(s) = \hat{f}(s)$, which implies that $\tilde{z}(s)$ is a solution of the DAE (10). □

Example 4.2. Consider an index-3 DAE

$$\begin{pmatrix} s & s & \alpha_1 \\ s & s & 0 \\ 0 & \alpha_2 & \alpha_3 s^2 - 2s + \alpha_4 \end{pmatrix} \begin{pmatrix} \tilde{x}_1(s) \\ \tilde{x}_2(s) \\ \tilde{x}_3(s) \end{pmatrix} = \begin{pmatrix} \hat{f}_1(s) \\ \hat{f}_2(s) \\ \hat{f}_3(s) \end{pmatrix}, \tag{13}$$

where $\alpha_1, \dots, \alpha_4$ are independent parameters. Following (12), we convert this DAE into

$$\left(\begin{array}{ccc|ccc} 1 & 0 & 0 & s & s & 0 \\ 0 & 1 & 0 & s & s & 0 \\ 0 & 0 & 1 & 0 & 0 & -2s \\ \hline -\tau_1 & 0 & 0 & 0 & 0 & \alpha_1' \\ 0 & -\tau_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\tau_3 & 0 & \alpha_2' & \alpha_3' s^2 + \alpha_4' \end{array}\right) \begin{pmatrix} \tilde{y}_1(s) \\ \tilde{y}_2(s) \\ \tilde{y}_3(s) \\ \tilde{z}_1(s) \\ \tilde{z}_2(s) \\ \tilde{z}_3(s) \end{pmatrix} = \begin{pmatrix} \hat{f}_1(s) \\ \hat{f}_2(s) \\ \hat{f}_3(s) \\ 0 \\ 0 \\ 0 \end{pmatrix}, \tag{14}$$

where τ_1, τ_2, τ_3 are independent parameters, $\alpha_1' = \tau_1 \alpha_1$ and $\alpha_i' = \tau_3 \alpha_i$ for $i = 2, 3, 4$. Then we can obtain a solution $(\tilde{x}_1(s), \tilde{x}_2(s), \tilde{x}_3(s))$ of (13) by solving the augmented DAE (14). While the index of (14) is also three, in general this conversion does not preserve the index of DAEs. □

After the index reduction process, we need to fill independent parameters by real numbers to start numerical methods. Indeed, we can substitute 1 for each diagonal entry τ_i of D, i.e., $D = I$. To explain this fact, let

$$B(s) = \begin{pmatrix} Q_1(s) & Q_2(s) \\ -D & DT(s) \end{pmatrix} \tag{15}$$

be the coefficient matrix of a DAE that our algorithm returns for the augmented DAE (12), where $Q_1(s)$ and $Q_2(s)$ are some polynomial matrices. By substituting the identity matrix to D, we obtain

$$\bar{B}(s) = \begin{pmatrix} Q_1(s) & Q_2(s) \\ -I & T(s) \end{pmatrix}. \tag{16}$$

Though $\bar{B}(s)$ is no longer an LM-polynomial matrix, the following lemma guarantees the upper-tightness of $\bar{B}(s)$.

Lemma 4.3. Let $Q_1(s)$, $Q_2(s)$ and $T(s)$ are polynomial matrices and D a nonsingular diagonal matrix. Then $B(s)$ in (15) is upper-tight if and only if $\bar{B}(s)$ in (16) is upper-tight.

Proof. Using $P = \begin{pmatrix} I & O \\ O & D^{-1} \end{pmatrix}$, we have $\bar{B}(s) = PB(s)$. Since P is a nonsingular constant matrix, $\delta_n(B) = \delta_n(\bar{B})$ holds. In addition, since P is nonsingular, diagonal and constant, the row transformation by P does not change the bipartite graph $G(B)$ and its edge weight c_e associated with $B(s)$. This fact implies that $\hat{\delta}_n(B) = \hat{\delta}_n(\bar{B})$. Thus the upper-tightness of $B(s)$ and $\bar{B}(s)$ are equivalent. □

From this lemma, we can "forget" the existence of D in the augmented DAE (12). That is, to reduce the index of the DAE (10), it suffices to apply our algorithm to a DAE

$$\begin{pmatrix} I & Q(s) \\ -I & T(s) \end{pmatrix} \begin{pmatrix} \tilde{y}(s) \\ \tilde{z}(s) \end{pmatrix} = \begin{pmatrix} \hat{f}(s) \\ 0 \end{pmatrix}$$

as if the set of nonzero coefficients of entries in $\begin{pmatrix} -I & T(s) \end{pmatrix}$ were independent.

4.2 Construction of Dual Optimal Solution

For an LM-polynomial matrix $A(s) = \binom{Q(s)}{T(s)}$, an optimal solution (p, q) of D(A) is constructed as follows. First, we obtain a maximum-weight perfect matching M in $G(A)$ by the Hungarian method [8]. Next, construct a residual graph $G_M = (W, E_M)$ with $W = R \cup C \cup \{r\}$ and $E_M = E(A) \cup M^\circ \cup Z$, where $R = \mathrm{Row}(A)$, $C = \mathrm{Col}(A)$, r is a new vertex, $M^\circ = \{(j, i) \mid (i, j) \in M\}$, and $Z = \{(r, i) \mid i \in R\}$. The arc length $\gamma\colon E_M \to \mathbb{Z}$ of G_M is defined by

$$\gamma(i, j) = \begin{cases} -c_{i,j} & ((i, j) \in E(A)), \\ c_{j,i} & ((i, j) \in M^\circ), \\ 0 & ((i, j) \in Z), \end{cases}$$

for each $(i, j) \in E_M$.

Lemma 4.4. For a residual graph G_M defined above, the following hold.

(1) All vertices are reachable from r on G_M.

(2) There is no negative-weight directed cycle with respect to the arc length γ in G_M.

Proof. (1) Every vertex $i \in R$ is reachable from r through an edge $(r, i) \in Z$. In addition, since $G(A)$ has a perfect matching M, every vertex $j \in C$ is also reachable from r via $i \in R$ through edges $(r, i) \in Z$ and $(i, j) \in M \subseteq E(A)$.

(2) Suppose that G_M has a simple directed cycle $\mathcal{C} \subseteq E_M$ of negative weight. By the construction of the residual graph G_M, we can express $\mathcal{C}$ as

$$\mathcal{C} = \{(i_0, j_1), (j_1, i_1), (i_1, j_2), \ldots, (i_{L-1}, j_L), (j_L, i_L)\},$$

where $L = |\mathcal{C}|/2$, $i_0, \ldots, i_L \in R$, $j_1, \ldots, j_L \in C$ and $i_0 = i_L$. Notice that $(i_{k-1}, j_k) \notin M$ and $(j_k, i_k) \in M$ hold for $k = 1, \ldots, L$. Let

$$M' := (M \setminus \{(j_1, i_1), \ldots, (j_L, i_L)\}) \cup \{(i_0, j_1), \ldots, (i_{L-1}, j_L)\}.$$

Now M' is a perfect matching of $G(A)$, and since

$$\sum_{e \in \mathcal{C}} \gamma(e) = -\sum_{k=1}^{L} c_{i_{k-1}, j_k} + \sum_{k=1}^{L} c_{j_k, i_k} < 0,$$

the weight of M' is greater than the weight of M. This contradicts the assumption that M is a maximum-weight perfect matching. □

This lemma guarantees the existence of a shortest path from r to each vertex with respect to the arc length γ in G_M. Let $d(i, j)$ be the length of a shortest path from $i \in W$ to $j \in W$. Using d, we define

$$p_i := -d(r, i) + \max_{i^* \in R} d(r, i^*), \tag{17}$$

$$q_j := -d(r, j) + \max_{i^* \in R} d(r, i^*) \tag{18}$$

for each $i \in R$ and $j \in C$.

Lemma 4.5. Let (p, q) be defined in (17) and (18). Then (p, q) is an optimal solution of $\mathrm{D}(A)$ satisfying $0 \le p_i \le ln$ for each $i \in R$ and $0 \le q_j \le ln$ for each $j \in C$, where $n = |R| = |C|$ and l is the maximum degree of an entry in $A(s)$.

Proof. First, we prove that (p, q) is a feasible solution of $\mathrm{D}(A)$. By the definition of (p, q), every p_i $(i \in R)$ and q_j $(j \in C)$ are clearly integer. For each $(i, j) \in E(A)$, it holds $d(r, j) \le d(r, i) - c_{i,j}$. Thus

$$q_j - p_i = -d(r, j) + d(r, i) \ge c_{i,j}$$

and this implies that (p, q) is a feasible solution of $\mathrm{D}(A)$.

We second show the optimality of (p, q). For each $(i, j) \in M$, since $(i, j) \in E_M$ and $(j, i) \in E_M$, we obtain

$$q_j - p_i = -d(r, j) + d(r, i) = c_{i,j}.$$

Thus it holds that

$$\sum_{j \in C} q_j - \sum_{i \in R} p_i = -\sum_{j \in C} d(r, j) + \sum_{i \in R} d(r, i) = \sum_{(i,j) \in M} (-d(r, j) + d(r, i)) = \sum_{(i,j) \in M} c_{i,j}$$

which implies that (p, q) is optimal to $\mathrm{D}(A)$.

Finally, we give the lower and upper bounds of each p_i and q_j. The non-negativity of p_i clearly holds from the definition of p_i. In addition, since $G(A)$ has a perfect matching, each $j \in C$ is incident to at least one vertex $i \in R$. Thus we obtain $q_j \ge p_i + c_{i,j} \ge 0$ by $p_i \ge 0$ and $c_{i,j} \ge 0$. Let $i^* \in R$ denote a vertex such that $d(r, i^*) \ge d(r, i)$ for each $i \in R$. Fix $j \in C$. Let $P_j \subseteq E_M$ and $P_{i^*} \subseteq E_M$ be shortest paths from r to j and i^*, respectively. Let $v \in W$ be the last common vertex in P_j and P_{i^*}. Note that it holds

$$q_j = -d(r, j) + d(r, i^*) = -d(v, j) + d(v, i^*).$$

Let $Q_j \subseteq P_j$ and $Q_{i^*} \subseteq P_{i^*}$ be subpaths from v to j and i^*, respectively. From the definition of the arc lengths, $-d(v,j) \le l|Q_j \cap E(A)|$ and $d(v,i^*) \le l|Q_{i^*} \cap M^\circ|$ hold. If $v = r$, we obtain

$$\begin{aligned} q_j &\le l(|Q_j \cap E(A)| + |Q_{i^*} \cap M^\circ|) \\ &= l\left(\frac{|Q_j|}{2} + \frac{|Q_{i^*}| - 1}{2}\right) \\ &< ln. \end{aligned}$$

The last inequality holds since $|Q_j| + |Q_{i^*}| \le \bigl|W \setminus \{r\}\bigr| = 2n$. Otherwise, it holds

$$\begin{aligned} q_j &\le l(|Q_j \cap E(A)| + |Q_{i^*} \cap M^\circ|) \\ &\le l\left(\left\lfloor\frac{|Q_j| + 1}{2}\right\rfloor + \left\lfloor\frac{|Q_{i^*}| + 1}{2}\right\rfloor\right) \\ &\le ln, \end{aligned}$$

where $|Q_j| + |Q_{i^*}| \le \bigl|W \setminus \{r, v\}\bigr| = 2n - 1$ is used in the last inequality. Thus $q_j \le ln$ holds for each $j \in C$. In addition, for each $i \in R$, we have $p_i \le q_j - c_{i,j} \le q_j \le ln$, where $j \in C$ is incident to i in M. □

4.3 Matrix Modification

Let $A(s) = \binom{Q(s)}{T(s)}$ be an $n \times n$ nonsingular LM-polynomial matrix that is not upper-tight. Let $A^\# = \binom{Q^\#}{T^\#}$ be the tight coefficient matrix with respect to an optimal solution (p, q) of $\mathrm{D}(A)$. Without loss of generality, we assume that $\mathrm{Row}(Q) = R_Q = \{1, \dots, m_Q\}$ and $p_1 \le \dots \le p_{m_Q}$, where $m_Q = |R_Q|$.

Recall the rank identity (9). Let $J^* \subseteq C$ be a column subset that minimizes the right-hand side of the identity for $A^\#$, i.e., it holds

$$\operatorname{rank} A^\# = \operatorname{rank} Q^\#[R_Q, J^*] + \text{t-rank}\, T^\#[R_T, J^*] + |C \setminus J^*|. \tag{19}$$

Such J^* is called a *minimizer* of (9). By the row transformation of $Q^\#$, we obtain a matrix $\bar{Q}^\# = UQ^\#$ such that

$$\operatorname{rank} \bar{Q}^\#[R_Q, J^*] = \text{t-rank}\, \bar{Q}^\#[R_Q, J^*]. \tag{20}$$

In particular, this transformation can be accomplished only by operations of adding a scalar multiple of a row $i \in R_Q$ to another row $j \in R_Q$ with $p_i > p_j$. Then the matrix U is upper-triangular due to the order of rows in R_Q. This is known as the *backward elimination* of $\bar{Q}^\#[R_Q, J^*]$. Consider

$$U_Q(s) = \operatorname{diag}(s^{-p_1}, \dots, s^{-p_{m_Q}}) U \operatorname{diag}(s^{p_1}, \dots, s^{p_{m_Q}}), \tag{21}$$

where $\operatorname{diag}(a_1, \dots, a_n)$ denotes a diagonal matrix with diagonal entries $a_1, \dots, a_n$. Note that each entry in $U_Q(s)$ is a polynomial because U is upper-triangular. In addition, since $\det U_Q(s) = \det U$ is a non-zero constant, $U_Q(s)$ is unimodular.

We define $D_p(s) = \operatorname{diag}(s^{p_1}, \dots, s^{p_n})$ and $D_q(s) = \operatorname{diag}(s^{q_1}, \dots, s^{q_n})$. Using $U_Q(s)$, we update $A(s)$ to $\bar{A}(s)$ as in (11):

$$\bar{A}(s) = \begin{pmatrix} U_Q(s) & O \\ O & I \end{pmatrix} A(s) = D_p^{-1}(s) \begin{pmatrix} U & O \\ O & I \end{pmatrix} D_p(s) A(s). \tag{22}$$

To show that (p, q) is not an optimal solution of $\mathrm{D}(\bar{A})$, we use the following lemma, which is given by Murota [12] as a combinatorial counterpart to Lemma 2.2.

Lemma 4.6 ([12, Proposition 6.2])**.** Let $A(s)$ be an $n \times n$ nonsingular polynomial matrix and $A^\#$ the tight coefficient matrix of $A(s)$ with respect to a feasible solution (p, q) of $\mathrm{D}(A)$. Then (p, q) is optimal if and only if $\text{t-rank}\, A^\# = n$.

Lemma 4.7. Let $A(s) = \binom{Q(s)}{T(s)}$ be an $n \times n$ nonsingular LM-polynomial matrix that is not upper-tight, and $A^\# = \binom{Q^\#}{T^\#}$ the tight coefficient matrix with respect to an optimal solution (p, q) of $\mathrm{D}(A)$. Then for the LM-polynomial matrix $\bar{A}(s)$ defined in (22), the value (p, q) is feasible on $\mathrm{D}(\bar{A})$ but not optimal.

Proof. Consider a rational function matrix

$$H(s) = D_p(s)\bar{A}(s)D_q^{-1}(s). \tag{23}$$

For each $i \in R$ and $j \in C$, it holds that $\deg H_{i,j}(s) = \bar{c}_{i,j} + p_i - q_j$, where $\bar{c}_{i,j} = \deg \bar{A}_{i,j}(s)$. By substituting (22) into (23), we obtain

$$H(s) = \begin{pmatrix} U & O \\ O & I \end{pmatrix} D_p(s)A(s)D_q^{-1}(s) = \begin{pmatrix} U & O \\ O & I \end{pmatrix} \bigl(A^{\#} + A^{\infty}(s)\bigr),$$

where $A^{\infty}(s)$ is a matrix whose nonzero entries are polynomials in s^{-1}. Hence for each $i \in R$ and $j \in C$, it holds $\deg H_{i,j}(s) \le 0$, which implies $\bar{c}_{i,j} \le q_j - p_i$. Therefore (p,q) is feasible on $\mathrm{D}\bigl(\bar{A}\bigr)$.

Next, we show that (p,q) is not optimal on $\mathrm{D}\bigl(\bar{A}\bigr)$. From (22), the tight coefficient matrix $\bar{A}^{\#}$ of $\bar{A}(s)$ with respect to (p,q) is

$$\bar{A}^{\#} = \begin{pmatrix} U & O \\ O & I \end{pmatrix} A^{\#} = \begin{pmatrix} \bar{Q}^{\#} \\ T^{\#} \end{pmatrix}, \tag{24}$$

where $\bar{Q}^{\#} = UQ^{\#}$. From Proposition 3.3 and (20), it holds

$$\begin{aligned} \text{t-rank}\, \bar{A}^{\#} &= \min\bigl\{\text{t-rank}\, \bar{Q}^{\#}[R_Q, J] + \text{t-rank}\, T^{\#}[R_T, J] + |C \setminus J| \bigm| J \subseteq C\bigr\} \\ &\le \text{t-rank}\, \bar{Q}^{\#}[R_Q, J^*] + \text{t-rank}\, T^{\#}[R_T, J^*] + |C \setminus J^*| \\ &= \operatorname{rank} \bar{Q}^{\#}[R_Q, J^*] + \text{t-rank}\, T^{\#}[R_T, J^*] + |C \setminus J^*|. \end{aligned}$$

Now since $Q^{\#}[R_Q, J^*]$ and $\bar{Q}^{\#}[R_Q, J^*] = UQ^{\#}[R_Q, J^*]$ have the same rank, we obtain

$$\text{t-rank}\, \bar{A}^{\#} \le \operatorname{rank} Q^{\#}[R_Q, J^*] + \text{t-rank}\, T^{\#}[R_T, J^*] + |C \setminus J^*| = \operatorname{rank} A^{\#},$$

where the last equality comes from (19). In addition, since $\operatorname{rank} \bar{A}^{\#} = \operatorname{rank} A^{\#}$ from (24), we have $\text{t-rank}\, \bar{A}^{\#} \le \operatorname{rank} \bar{A}^{\#}$, which implies $\text{t-rank}\, \bar{A}^{\#} = \operatorname{rank} \bar{A}^{\#} = \operatorname{rank} A^{\#}$. Furthermore, since $A(s)$ is not upper-tight, we have $\operatorname{rank} A^{\#} < n$ by Lemma 2.2. Thus, $\text{t-rank}\, A^{\#} = \operatorname{rank} A^{\#} < n$ holds. It then follows from Lemma 4.6 that (p,q) is not optimal on $\mathrm{D}\bigl(\bar{A}\bigr)$. $\square$

From Lemma 4.7 and the unimodularity of $U_Q(s)$, we obtain the following.

Corollary 4.8. Let $A(s) = \binom{Q(s)}{T(s)}$ be an $n \times n$ nonsingular LM-polynomial matrix that is not upper-tight, and $\bar{A}(s)$ the LM-polynomial matrix defined in (22). Then $\hat{\delta}_n\bigl(\bar{A}\bigr) \le \hat{\delta}_n(A) - 1$ and $\delta_n(A) = \delta_n\bigl(\bar{A}\bigr)$ hold.

4.4 Dual Updates

Let (p,q) be a feasible solution of $\mathrm{D}\bigl(\bar{A}\bigr)$. We obtain an optimal solution of $\mathrm{D}\bigl(\bar{A}\bigr)$ by iterating the following procedure.

Let $\bar{A}^{\#}$ be the tight coefficient matrix of $\bar{A}(s)$ with respect to (p,q). First we check $\text{t-rank}\, \bar{A}^{\#} = n$ or not. If it is, (p,q) is an optimal solution of $\mathrm{D}\bigl(\bar{A}\bigr)$ from Lemma 4.6 and we are done. Otherwise, we construct a feasible solution (p',q') of $\mathrm{D}\bigl(\bar{A}\bigr)$ such that the difference

$$\Delta \coloneqq \sum_{j \in C} \bigl(q'_j - q_j\bigr) - \sum_{i \in R} (p'_i - p_i) \tag{25}$$

of the objective values is negative. Let $G^{\#} = \bigl(R, C; E^{\#}\bigr)$ be a bipartite graph defined by

$$E^{\#} \coloneqq \bigl\{(i,j) \in R \times C \bigm| \bar{A}^{\#}_{i,j} \neq 0\bigr\} = \bigl\{(i,j) \in E\bigl(\bar{A}\bigr) \bigm| q_j - p_i = \bar{c}_{i,j}\bigr\}.$$

Since (p,q) is not optimal, there is no perfect matching of $G^{\#}$. Thus $G^{\#}$ has a vertex cover $S \subseteq R \cup C$ with $|S| < n$ by König–Egerváry's theorem. Using this S, we define (p',q') as follows:

$$p'_i = \begin{cases} p_i & (i \in R \cap S) \\ p_i + 1 & (i \in R \setminus S) \end{cases}, \quad q'_j = \begin{cases} q_j + 1 & (j \in C \cap S) \\ q_j & (j \in C \setminus S) \end{cases} \tag{26}$$

for $i \in R$ and $j \in C$.

Lemma 4.9. Let (p, q) be a feasible but not optimal solution of $\mathrm{D}(\bar{A})$ and (p', q') defined in (26). Then the difference Δ of the objective values in (25) is negative, and (p', q') is a feasible solution of $\mathrm{D}(\bar{A})$.

Proof. The difference of the objective values is $\Delta = |C \cap S| - |R \setminus S| = |S| - |R| < 0$. Next, we show the feasibility of (p', q'). For every $(i, j) \in E(\bar{A})$, it holds $q_j - p_i \geq \bar{c}_{i,j}$ since (p, q) is feasible. If $i \in S$ or $j \in S$, it holds $(q'_j - q_j) - (p'_i - p_i) \geq 0$, which imply $q'_j - p'_i \geq q_j - p_i \geq \bar{c}_{i,j}$. If $i \notin W$ and $j \notin W$, then (i, j) is not an edge of $G^{\#}$ since (i, j) is not covered by W. Hence it holds $q_j - p_i > \bar{c}_{i,j}$, and thus $q'_j - p'_i = q_j - p_i - 1 \geq \bar{c}_{i,j}$. □

We update (p, q) to (p', q'), and go back to the optimality checking. From Lemma 4.9, it is guaranteed that (p, q) becomes an optimal solution of $\mathrm{D}(\bar{A})$ by iterating the update process above.

4.5 Complexity Analysis

This section is devoted to complexity analysis. The dominating part in our algorithm is the matrix multiplications in (22).

Let $A(s)$ be an $n \times n$ nonsingular LM-polynomial matrix and $A^{\#}$ the tight coefficient matrix with respect to an optimal solution (p, q) of $\mathrm{D}(A)$. From the definition of $A^{\#}$, we can express $A(s)$ as

$$A(s) = D_p^{-1}(s) \left(A^{\#} + \sum_{k=1}^{K} s^{-k} V_k \right) D_q(s) \tag{27}$$

for some K matrices $V_1, V_2, \ldots, V_K$ with $V_K \neq O$. By (22) and (27), we have

$$\bar{A}(s) = D_p^{-1}(s) \begin{pmatrix} U & O \\ O & I \end{pmatrix} \left(A^{\#} + \sum_{k=1}^{K} s^{-k} V_k \right) D_q(s).$$

Therefore, we can compute $\bar{A}(s)$ by performing $K + 1$ constant matrix multiplications.

By $V_K \neq O$, there exist $i \in R$ and $j \in C$ such that the (i, j) entry in V_K is nonzero. Then the degree of the corresponding term in $A_{i,j}(s)$ is equal to $q_j - p_i - K$. Since $A_{i,j}(s)$ is a polynomial, we have $q_j - p_i - K \geq 0$, which implies $K \leq q_j - p_i \leq q_j$. The following lemma bounds p_i and q_j at any iteration of our algorithm.

Lemma 4.10. During the algorithm, the values p_i and q_j are at most $2ln$ for $i \in R$ and $j \in C$, where l is the maximum degree of an entry in $A(s)$.

Proof. From Lemma 4.5, the initial values of p_i and q_j are bounded by ln. In every update of (p, q), the values p_i and q_j increases by at most one from the update rule (26). In addition, (p, q) is updated at most $\hat{\delta}_n(A) - \delta_n(A) \leq ln$ times because the objective value $\sum_{j \in C} q_j - \sum_{i \in R} p_i$ of the dual problem decreases by at least one in every update. Therefore, at any iteration of the algorithm, it holds $p_i, q_j \leq ln + \hat{\delta}_n(A) \leq ln + ln = 2ln$. □

The time complexity of our algorithm is as follows.

Theorem 4.11. Let $A(s)$ be an $n \times n$ nonsingular LM-polynomial matrix and l the maximum degree of an entry in $A(s)$. Then Algorithm for Tightness runs in $\mathrm{O}(l^2 n^{\omega+2})$ time, where $2 < \omega \leq 3$ is the matrix multiplication exponent.

Proof. Phase 1 can be done in $\mathrm{O}(n^3)$ time by the Hungarian method [8] and shortest path algorithms such as the Bellman–Ford algorithm. Consider the time complexity in every iteration of Phases 2 and 3. In Phase 2, the nonsingularity of the tight coefficient matrix $A^{\#}$ can be checked via the rank identity (9). Thus an efficient way is to obtain a minimizer J^* of (9) before Phase 2, and then check the nonsingularity of $A^{\#}$ by (9). The minimizer J^* can be found from a residual graph constructed by an augmenting path type algorithm [19], which runs in $\mathrm{O}(n^3 \log n)$ time [3]. The computation of $\bar{A}(s)$ in Phase 3 can be done in $\mathrm{O}(Nn^{\omega}) = \mathrm{O}(\max_{j \in C} q_j n^{\omega}) = \mathrm{O}(ln^{\omega+1})$ time from Lemma 4.10, where (p, q) is a dual optimal solution of $\mathrm{D}(A)$ and N is in (27). In addition, since the number of iterations of Phases 2 and 3 is at most $\hat{\delta}_n(A) - \delta_n(A) \leq ln$, the running time in Phases 2 and 3 is $\mathrm{O}(l^2 n^{\omega+2})$. Finally, the updates of (p, q) run in $\mathrm{O}(ln^4)$ time: (p, q) is updated at most $\hat{\delta}_n(A) \leq ln$ times, and in every update, we can find a vertex cover in $\mathrm{O}(n^3)$ time by Ford–Fulkerson's algorithm. Thus the total running time is $\mathrm{O}(l^2 n^{\omega+2})$. □

Theorem 4.12. For a DAE (10) with $n \times n$ nonsingular mixed polynomial matrix $A(s)$, our algorithm returns an equivalent DAE of index zero or one in $\mathrm{O}(l^2 n^{\omega+2})$ time, where $2 < \omega \leq 3$ is the matrix multiplication exponent and l is the maximum degree of entries in $A(s)$.

Proof. First we convert the DAE into an equivalent DAE with LM-polynomial matrix $A^{\mathrm{LM}}(s)$ of size $2n \times 2n$. Note that the maximum degree of an entry in $A^{\mathrm{LM}}(s)$ is equal to l by (12). Hence it follows from Theorem 4.11 that Algorithm for Tightness for $A^{\mathrm{LM}}(s)$ runs in $\mathrm{O}\bigl(l^2 n^{\omega+2}\bigr)$ time. The resulting DAE has a coefficient matrix such that the maximum degree of an entry is at most $4ln$, because it holds that

$$\deg A^{\mathrm{LM}}_{i,j}(s) \le q_j - p_i \le q_j \le 4ln$$

with a feasible solution (p, q) of $\mathrm{D}\bigl(A^{\mathrm{LM}}\bigr)$, where the last inequality is due to Lemma 4.10.

Next we analyze the complexity of the MS-algorithm described in Section 2.4. In Step 1, we can reuse a dual optimal solution (p, q) obtained at the termination of Algorithm for Tightness, or compute a new $(\tilde{p}, \tilde{q})$ such that $\tilde{p}_i \le p_i$ for $i \in R$ to decrease the number of dummy variables, in $\mathrm{O}\bigl(n^3\bigr)$ time. The nonsingularity of the corresponding tight coefficient matrix can be verified by solving an independent matching problem in $\mathrm{O}\bigl(n^3 \log n\bigr)$ time [3, 19]. Step 2 runs in $\mathrm{O}\bigl(n^4 \log n\bigr)$ time since we solve independent matching problems at most $2n$ times. We now consider the resultant DAE returned in Step 4. The number of original (non-dummy) variables is $2n$, and from Lemma 4.10, the orders of their derivatives are at most

$$4ln + \max_{i \in R} p_i \le 4ln + 4ln = \mathrm{O}(ln).$$

In contrast, the number of dummy variables is $\sum_{i \in R} p_i = \mathrm{O}\bigl(ln^2\bigr)$, and there is no derivative of dummy variables in the resultant DAE. Therefore, the number of terms in the resultant DAE is $\mathrm{O}\bigl(ln^2\bigr)$, and thus Step 4 runs in $\mathrm{O}\bigl(ln^2\bigr)$ time. Therefore the MS-algorithm costs $\mathrm{O}\bigl(n^4 \log n + ln^2\bigr)$ time.

Since the bottleneck in the entire algorithm is Algorithm for Tightness, the total running time of our algorithm is $\mathrm{O}\bigl(l^2 n^{\omega+2}\bigr)$. □

5 Exploiting Dimensional Consistency

5.1 Dimensional Consistency

The principle of dimensional homogeneity claims that any equation describing a physical phenomenon must be consistent with respect to physical dimensions. In order to reflect the dimensional consistency in conservation laws of dynamical systems, Murota [11] introduced a class of mixed polynomial matrices $A(s) = Q(s) + T(s)$ that satisfy the following condition.

(MP-DC) Every nonvanishing subdeterminant of $Q(s)$ is a monomial in s.

A mixed polynomial matrix satisfying (MP-DC) is said to be *dimensionally consistent*. We abbreviate a dimensionally consistent mixed polynomial matrix and a dimensionally consistent LM-polynomial matrix to a *DCM-polynomial matrix* and a *DCLM-polynomial matrix*, respectively. It is known that an $m \times n$ polynomial matrix $Q(s)$ satisfies (MP-DC) if and only if $Q(s)$ is written as

$$Q(s) = \mathrm{diag}(s^{-\lambda_1}, \ldots, s^{-\lambda_m})Q(1)\mathrm{diag}(s^{\mu_1}, \ldots, s^{\mu_n}) \tag{28}$$

for some integers $\lambda_1, \ldots, \lambda_m$ and $\mu_1, \ldots, \mu_n$.

5.2 Improved Algorithm

This section improves the matrix modification procedure in Phase 3 for DCLM-polynomial matrices preserving their dimensional consistency.

Let $A(s) = \binom{Q(s)}{T(s)}$ be a DCLM-polynomial matrix with $R_Q = \mathrm{Row}(Q)$, $R_T = \mathrm{Row}(T)$ and $C = \mathrm{Col}(A)$. Let $A^{\#} = \binom{Q^{\#}}{T^{\#}}$ denote the tight coefficient matrix with respect to an optimal solution (p, q) of $\mathrm{D}(A)$. For an integer $k \in \mathbb{Z}$, let

$$R_k = \{i \in R_Q \mid p_i - \lambda_i = k\}, \quad C_k = \{j \in C \mid q_j - \mu_j = k\}. \tag{29}$$

If $Q^{\#}_{i,j} \ne 0$, then we have $c_{i,j} = q_j - p_i$ from the definition of $A^{\#}$ and $c_{i,j} = \mu_j - \lambda_i$ by (28). Hence $q_j - p_i = \mu_j - \lambda_i$ follows, which implies $i \in R_k$ if and only if $j \in C_k$. Thus, it holds $A^{\#}[R_h, C_k] = O$ for distinct $h, k \in \mathbb{Z}$. Namely, $Q^{\#}$ forms a block diagonal matrix as

$$Q^{\#} = \begin{array}{c} \\ \vdots \\ R_{-1} \\ R_0 \\ R_1 \\ R_2 \\ \vdots \end{array} \begin{array}{c} \begin{array}{cccccc} \cdots & C_{-1} & C_0 & C_1 & C_2 & \cdots \end{array} \\ \begin{pmatrix} \ddots & & & & & \\ & Q^{\#}_{-1} & & & & \\ & & Q^{\#}_0 & & & \\ & & & Q^{\#}_1 & & \\ & & & & Q^{\#}_2 & \\ & & & & & \ddots \end{pmatrix} \end{array},$$

where $Q^{\#}_k = Q^{\#}[R_k, C_k]$ for $k \in \mathbb{Z}$, and empty blocks indicate zero submatrices.

Let $J^* \subseteq C$ be a minimizer of the rank identity (9) for $A^{\#}$. Sorting rows in ascending order of p, the matrix modification process described in Section 4.3 finds a nonsingular upper-triangular matrix U such that

$$\operatorname{rank} UQ^{\#}[R_Q, J^*] = \text{t-rank}\, UQ^{\#}[R_Q, J^*]. \tag{30}$$

For a DCLM-polynomial matrix, supposing that rows in R_k is sorted in ascending order of p, we find a nonsingular upper-triangular matrix U_k such that

$$\operatorname{rank} U_k Q^{\#}_k[R_k, C_k \cap J^*] = \text{t-rank}\, U_k Q^{\#}_k[R_k, C_k \cap J^*]$$

for $k \in \mathbb{Z}$. Then $U = \text{block-diag}(\ldots, U_{-1}, U_0, U_1, U_2, \ldots)$ satisfies (30), where $\text{block-diag}(B_1, B_2, \ldots, B_N)$ is a block diagonal matrix of diagonal blocks $B_1, B_2, \ldots, B_N$.

For $k \in \mathbb{Z}$, let $P_k(s)$ be a diagonal polynomial matrix with $\text{Row}(P_k) = \text{Col}(P_k) = R_k$ whose (i,i) entry is s^{p_i} for each $i \in R_k$. Let $D_p(s) = \text{block-diag}(\ldots, P_{-1}(s), P_0(s), P_1(s), P_2(s), \ldots)$. Now the unimodular matrix $U_Q(s)$ defined in (21) can be written as

$$\begin{aligned} U_Q(s) &= D_p^{-1}(s)\text{block-diag}(\ldots, U_{-1}, U_0, U_1, U_2, \ldots)D_p(s) \\ &= D_p^{-1}(s)\text{block-diag}(\ldots, U_{-1}P_{-1}(s), U_0P_0(s), U_1P_1(s), U_2P_2(s), \ldots). \end{aligned} \tag{31}$$

Then we update $A(s)$ into $\bar{A}(s) = \binom{U_Q(s)Q(s)}{T(s)}$ as written in (22).

Lemma 5.1. Let $A(s) = \binom{Q(s)}{T(s)}$ be an $n \times n$ DCLM-polynomial matrix. Then $\bar{A}(s) = \binom{U_Q(s)Q(s)}{T(s)}$ is also dimensionally consistent.

Proof. Let $\lambda_1, \ldots, \lambda_{m_Q}$ and $\mu_1, \ldots, \mu_n$ defined in (28) for $A(s)$, where $m_Q = |\text{Row}(Q)|$. For $k \in \mathbb{Z}$, let R_k and C_k defined in (29), and let $\Lambda_k(s)$ denote a diagonal polynomial matrix with $\text{Row}(\Lambda_k) = \text{Col}(\Lambda_k) = R_k$ whose (i,i) entry is s^{λ_i} for each $i \in R_k$, and $D_\mu(s) = \text{diag}(s^{\mu_1}, \ldots, s^{\mu_n})$. Then the condition (28) for dimensional consistency is written as

$$Q(s) = \text{block-diag}(\ldots, \Lambda_{-1}^{-1}(s), \Lambda_0^{-1}(s), \Lambda_1^{-1}(s), \Lambda_2^{-1}(s), \ldots)Q(1)D_\mu(s). \tag{32}$$

Combining (31) and (32), we obtain

$$\begin{aligned} U_Q(s)Q(s) &= P^{-1}(s)\text{block-diag}(\ldots, U_{-1}P_{-1}(s)\Lambda_{-1}^{-1}(s), U_0P_0(s)\Lambda_0^{-1}(s), U_1P_1(s)\Lambda_1^{-1}(s), \ldots)Q(1)D_\mu(s) \\ &= P^{-1}(s)\text{block-diag}(\ldots, s^{-1}U_{-1}, U_0, sU_1, s^2U_2, \ldots)Q(1)D_\mu(s) \\ &= \text{block-diag}(\ldots, s^{-1}P_{-1}^{-1}(s), P_0^{-1}(s), sP_1^{-1}(s), s^2P_2^{-1}(s), \ldots)UQ(1)D_\mu(s), \end{aligned} \tag{33}$$

where we used $P_k(s)\Lambda_k^{-1}(s) = s^kI$ for $k \in \mathbb{Z}$. From (33), $\bar{A}(s)$ is also dimensionally consistent. □

5.3 Complexity Analysis

For a DCLM-polynomial matrix $A(s)$, we can compute $\bar{A}(s) = U(s)A(s)$ only by one constant matrix multiplication $UQ(1)$ from (33), whereas a general LM-polynomial matrix needs $\mathrm{O}(ln)$ multiplications. This improves the total running time as follows.

Theorem 5.2. Let $A(s)$ be an $n \times n$ nonsingular DCLM-polynomial matrix and l the maximum degree of an entry in $A(s)$. Then Algorithm for Tightness runs in $\mathrm{O}(ln^4 \log n)$ time.

Proof. For each iteration of Phases 2 and 3, the computation of $\bar{A}(s)$ in Phase 3 can be done in $\mathrm{O}(n^{\omega})$ time, where $2 < \omega \leq 3$ is the matrix multiplication exponent. The most expensive part is the nonsingularity checking for a tight coefficient matrix in Phase 2, which requires $\mathrm{O}\big(n^3 \log n\big)$ time [3, 19]. Since the number of iterations of Phases 2 and 3 is at most $\hat{\delta}_n(A) - \delta_n(A) \leq ln$, the running time of Phases 2 and 3 is $\mathrm{O}\big(ln^4 \log n\big)$. We can check that other computations run in $\mathrm{O}\big(ln^4 \log n\big)$ time as in the proof of Theorem 4.11. □

Theorem 5.3. For a DAE (10) with $n \times n$ nonsingular DCM-polynomial coefficient matrix $A(s)$, our algorithm returns an equivalent DAE of index zero or one in $\mathrm{O}\big(ln^4 \log n\big)$ time, where l is the maximum degree of entries in $A(s)$.

Proof. We can easily check that the coefficient LM-polynomial matrix of the augmented DAE described in Section 4.1 is also dimensionally consistent. Algorithm for Tightness runs in $\mathrm{O}\big(ln^4 \log n\big)$ time from Theorem 5.2. In addition, the MS-algorithm runs in $\mathrm{O}\big(n^4 \log n + ln^2\big)$ time as discussed in the proof of Theorem 4.12. Thus the total running time is $\mathrm{O}\big(ln^4 \log n\big)$. □

6 Examples

We give two examples below. The first example is a simple index-4 DAE and the second example is a DAE representing an electrical network. Throughout the execution of our algorithm, it is emphasized that: (i) we only use combinatorial operations and numerical calculations over rational numbers (especially over integers in the following examples), and (ii) we do not reference nominal values of physical quantities.

6.1 Example of High-index DAE

The first example is the following index-4 DAE

$$\begin{cases} \ddot{x}_1 - \dot{x}_1 + \ddot{x}_2 - \dot{x}_2 + x_4 = f_1(t), \\ \ddot{x}_1 + \ddot{x}_2 + x_3 = f_2(t), \\ \alpha_1 x_2 + \alpha_2 \ddot{x}_3 + \alpha_3 \dot{x}_4 = f_3(t), \\ \alpha_4 x_3 + \alpha_5 \dot{x}_4 = f_4(t), \end{cases} \tag{34}$$

with independent parameters $\alpha_1, \ldots, \alpha_5$ and smooth functions $f_1, \ldots, f_4$. The coefficient matrix $A(s) = \binom{Q(s)}{T(s)}$ corresponding to (34) is an LM-polynomial matrix given by

$$A(s) = \begin{pmatrix} s^2 - s & s^2 - s & & 1 \\ s^2 & s^2 & 1 & \\ & \alpha_1 & \alpha_2 s^2 & \alpha_3 s \\ & & \alpha_4 & \alpha_5 s \end{pmatrix}, \tag{35}$$

where empty cells indicate zero. The row sets R_Q of $Q(s)$ and R_T of $T(s)$ correspond to the first and last two rows in $A(s)$, respectively. Since $\delta_n(A) = \deg\big(-\alpha_1\alpha_5 s^3 - \alpha_1\alpha_4 s^2 + \alpha_1\alpha_5 s^2\big) = 3$ and $\hat{\delta}_n(A) = 7$, the MS-algorithm is not applicable to the DAE, which is shown in our algorithm.

Let us apply our algorithm to (35). First, we find a dual optimal solution $p = (0, 0, 0, 0)$ and $q = (2, 2, 2, 1)$. The corresponding tight coefficient matrix $A^{\#} = \binom{Q^{\#}}{T^{\#}}$ is

$$A^{\#} = \begin{pmatrix} 1 & 1 & & \\ 1 & 1 & & \\ & & \alpha_2 & \alpha_3 \\ & & & \alpha_5 \end{pmatrix}.$$

A minimizer J^* of (9) for $A^{\#}$ is a set of the first two columns as follows:

$$A^{\#} = \begin{array}{cc} \begin{array}{cccc} \multicolumn{2}{c}{\overbrace{\hphantom{1\quad 1}}^{J^*}} & \multicolumn{2}{c}{\overbrace{\hphantom{\alpha_2\quad \alpha_3}}^{C\setminus J^*}} \end{array} & \\ \begin{pmatrix} 1 & 1 & & \\ 1 & 1 & & \\ & & \alpha_2 & \alpha_3 \\ & & & \alpha_5 \end{pmatrix} & \begin{matrix} \left.\vphantom{\begin{matrix}1\\1\end{matrix}}\right\} R_Q \\ \left.\vphantom{\begin{matrix}1\\1\end{matrix}}\right\} R_T \end{matrix} \end{array}.$$

Then we can check that $\operatorname{rank} A^{\#} = Q^{\#}[R_Q, J^*] + T^{\#}[R_T, J^*] + |C \setminus J^*| = 1 + 0 + 2 = 3 < 4$, which implies that $A(s)$ is not upper-tight. We convert $Q^{\#}[R_Q, J^*] = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ by the backward elimination into

$$\bar{Q}^{\#}[R_Q, J^*] = UQ^{\#}[R_Q, J^*] = \begin{pmatrix} & \\ 1 & 1 \end{pmatrix},$$

where $U = \begin{pmatrix} 1 & -1 \\ & 1 \end{pmatrix}$. Using $U_Q(s) = U$, the LM-polynomial matrix $A(s)$ is modified to

$$A'(s) = \begin{pmatrix} 1 & -1 & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{pmatrix} A(s) = \begin{pmatrix} -s & -s & -1 & 1 \\ s^2 & s^2 & 1 & \\ & \alpha_1 & \alpha_2 s^2 & \alpha_3 s \\ & & \alpha_4 & \alpha_5 s \end{pmatrix}.$$

The dual solution is updated to $p' = (1, 0, 0, 1)$ and $q' = (2, 2, 2, 2)$, and the corresponding tight coefficient matrix $A'^{\#} = \binom{Q'^{\#}}{T'^{\#}}$ of $A'(s)$ is

$$A'^{\#} = \begin{pmatrix} -1 & -1 & & \\ 1 & 1 & & \\ & & \alpha_2 & \\ & & & \alpha_5 \end{pmatrix}.$$

The minimizer J^* that we used above also minimizes the right-hand side of the rank identity (9) for $A'^{\#}$. Since $A'^{\#}$ is still singular, we go on the modification. Noting the order of rows, we transform $Q'^{\#}[R_Q, J^*] = \begin{pmatrix} -1 & -1 \\ 1 & 1 \end{pmatrix}$ by $U' = \begin{pmatrix} 1 & \\ 1 & 1 \end{pmatrix}$ into

$$\bar{Q}'^{\#}[R_Q, J^*] = U'Q'^{\#}[R_Q, J^*] = \begin{pmatrix} -1 & -1 \\ & \end{pmatrix}.$$

We have $U'_Q(s) = \operatorname{diag}(s^{-1}, 1)U'\operatorname{diag}(s, 1) = \begin{pmatrix} 1 & \\ s & 1 \end{pmatrix}$, and modify $A'(s)$ to

$$A''(s) = \begin{pmatrix} 1 & & & \\ s & 1 & & \\ & & 1 & \\ & & & 1 \end{pmatrix} A'(s) = \begin{pmatrix} -s & -s & -1 & 1 \\ & & -s+1 & s \\ & \alpha_1 & \alpha_2 s^2 & \alpha_3 s \\ & & \alpha_4 & \alpha_5 s \end{pmatrix}.$$

The dual solution is updated to $p'' = (1, 3, 2, 3)$ and $q'' = (2, 2, 4, 4)$. Our algorithm halts at this point since $A''(s)$ is upper-tight, which can be checked through the nonsingularity of the tight coefficient matrix $A''^{\#}$ again. Now $\delta_n(A)$ is computed as $\delta_n(A) = \delta_n(A'') = \hat{\delta}_n(A'') = 3$. The resulting DAE is

$$\begin{cases} -\dot{x}_1 - \dot{x}_2 - x_3 + x_4 = f_1(t) - f_2(t), \\ -\dot{x}_3 + x_3 + \dot{x}_4 = \dot{f}_1(t) - \dot{f}_2(t) + f_2(t), \\ \alpha_1 x_2 + \alpha_2 \ddot{x}_3 + \alpha_3 \dot{x}_4 = f_3(t), \\ \alpha_4 x_3 + \alpha_5 \dot{x}_4 = f_4(t), \end{cases} \tag{36}$$

which is index two.

An index-1 DAE is obtained by applying the MS-algorithm to the DAE (36). Instead of (p'', q''), we now use an optimal solution $\tilde{p} = (0, 2, 1, 2)$ and $\tilde{q} = (1, 1, 3, 3)$ of $\mathrm{D}(A'')$ in the MS-algorithm in order to decrease the number of dummy variables and equations. Then the MS-algorithm outputs an index-1

DAE

$$\begin{cases} -\dot{x}_1 - z_2^{[1]} - x_3 + x_4 = f_1(t) - f_2(t), \\ -\dot{x}_3 + x_3 + \dot{x}_4 = \dot{f}_1(t) - \dot{f}_2(t) + f_2(t), \\ -z_3^{[2]} + x_3 + z_4^{[2]} = \ddot{f}_1(t) - \ddot{f}_2(t) + \dot{f}_2(t), \\ -z_3^{[3]} + x_3 + z_4^{[3]} = \dddot{f}_1(t) - \dddot{f}_2(t) + \ddot{f}_2(t), \\ \alpha_1 x_2 + \alpha_2 x_3 + \alpha_3 \dot{x}_4 = f_3(t), \\ \alpha_1 z_2^{[1]} + \alpha_2 \dot{x}_3 + \alpha_3 z_4^{[2]} = \dot{f}_3(t), \\ \alpha_4 x_3 + \alpha_5 \dot{x}_4 = f_4(t), \\ \alpha_4 \dot{x}_3 + \alpha_5 z_4^{[2]} = \dot{f}_4(t), \\ \alpha_4 z_3^{[2]} + \alpha_5 z_4^{[3]} = \ddot{f}_4(t), \end{cases}$$

where $z_2^{[1]}$, $z_3^{[2]}$, $z_3^{[3]}$, $z_4^{[2]}$ and $z_4^{[3]}$ are dummy variables corresponding to $\dot{x}_2$, $\ddot{x}_3$, $\dddot{x}_3$, $\ddot{x}_4$ and $\dddot{x}_4$, respectively.

6.2 Example of Electrical Network

The next example is a DAE representing an electrical network illustrated in Figure 1, given in [17, Section 1.1]. The network consists of a voltage source of time-varying voltage $V(t)$, two resistances R_1 and R_2, an inductor L and a capacitor C. State variables of this network is currents $i_1, \dots, i_5$ shown in Figure 1, and voltages $v_1, \dots, v_5$ across branches carrying the currents $i_1, \dots, i_5$, respectively.

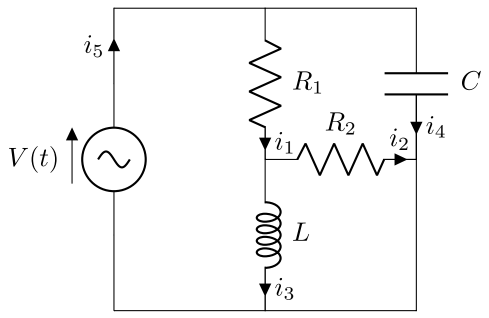

Figure 1: Simple RLC network.

The Laplace transform of an index-2 DAE representing this network is given by

$$\left(\begin{array}{ccccc|ccccc} -1 & & & -1 & 1 & & & & & \\ & 1 & 1 & 1 & -1 & & & & & \\ \hline & & & & & 1 & & 1 & & -1 \\ & & & & & -1 & -1 & & 1 & \\ & & & & & & 1 & -1 & & \\ \hline R_1 & & & & & -1 & & & & \\ & R_2 & & & & & -1 & & & \\ & & sL & & & & & -1 & & \\ & & & -1 & & & & & sC & \\ & & & & & & & & & 1 \end{array}\right) \begin{pmatrix} \tilde{i}_1 \\ \tilde{i}_2 \\ \tilde{i}_3 \\ \tilde{i}_4 \\ \tilde{i}_5 \\ \tilde{v}_1 \\ \tilde{v}_2 \\ \tilde{v}_3 \\ \tilde{v}_4 \\ \tilde{v}_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \tilde{V}(s) \end{pmatrix}, \qquad (37)$$

where $\tilde{i}_1, \dots, \tilde{i}_5$, $\tilde{v}_1, \dots, \tilde{v}_5$ and $\tilde{V}(s)$ are the Laplace transforms of $i_1, \dots, i_5$, $v_1, \dots, v_5$ and $V(t)$, respectively (we assumed that all state variables and their derivatives were equal to zero at $t = 0$ for simplicity). In this system (37), the first two equations come from Kirchhoff's current law (KCL), and the following three equations come from Kirchhoff's voltage law (KVL). The last five equations represent the element characteristics (constitutive equations). The coefficient matrix in (37) is naturally regarded as a mixed polynomial matrix with independent parameters R_1, R_2, L and C since nominal values of the parameters are supposed to be inaccurate.

Let $A(s)$ denote the coefficient matrix in (37). Since $A(s)$ is not LM-polynomial, it seems that we cannot directly apply our algorithm to $A(s)$. However, since each of the last five rows in $A(s)$ do not contain two or more accurate constants, we can convert $A(s)$ into an LM-polynomial matrix by multiplying an independent parameter to each of the rows. In addition, by the same logic to Lemma 4.3, our algorithm works without actually multiplying the independent parameters by regarding nonzero entries in the last five rows as independent parameters. Thus we see $A(s)$ as an LM-polynomial matrix $A(s) = \binom{Q(s)}{T(s)}$, where $Q(s)$ and $T(s)$ correspond to the first and last five rows in $A(s)$, respectively. In addition, $A(s)$ meets the condition (28) for DCLM-polynomial matrices with $\lambda = (0,0,0,0,0)$ and $\mu = (0,0,0,0,0,0,0,0,0,0)$.

We are now ready for applying our algorithm to $A(s)$. In Phase 1, a dual optimal solution is obtained as $p = (0,0,0,0,0,0,0,0,0,0)$ and $q = (0,0,1,0,0,0,0,0,0,1,0)$, which implies that $\hat{\delta}_n(A) = 2$. The corresponding tight coefficient matrix $A^\# = \binom{Q^\#}{T^\#}$ is given by

$$A^\# = \left(\begin{array}{ccccc|ccccc}
-1 & & & -1 & 1 & & & & & \\
 & 1 & & 1 & -1 & & & & & \\
\hline
 & & & & & 1 & & 1 & & -1 \\
 & & & & & -1 & -1 & & & \\
 & & & & & & 1 & -1 & & \\
\hline
R_1 & & & & & -1 & & & & \\
 & R_2 & & & & & -1 & & & \\
 & & L & & & & & -1 & & \\
 & & & -1 & & & & & C & \\
 & & & & & & & & & 1
\end{array}\right)\begin{array}{l}\left.\begin{array}{c}\\ \\ \\ \\ \\ \end{array}\right\} R_Q \\ \left.\begin{array}{c}\\ \\ \\ \\ \\ \end{array}\right\} R_T\end{array}.$$

A minimizer J^* of the rank identity (9) for $A^\#$ is the set of nine columns other than the rightmost column corresponding to the variable $\tilde{v}_5$. Thus we can check

$$\operatorname{rank} A^\# = Q^\#[R_Q, J^*] + T^\#[R_T, J^*] + |C \setminus J^*| = 4 + 4 + 1 = 9 < 10,$$

which imply that $A(s)$ is not upper-tight. We proceed to the matrix modification process for DCLM-polynomial matrices that we described in Section 5.2.

The row set R_k and the column set C_k for $k \in \mathbb{Z}$ defined in (29) is the following:

$$Q^\# = \begin{array}{c}
\begin{array}{cccccccccc}
\multicolumn{9}{c}{\overbrace{\hphantom{-1 \quad 1 \quad\; \quad -1 \quad 1 \quad 1 \quad -1 \quad 1 \quad\;}}^{J^*}} & \\
\multicolumn{2}{c}{\overbrace{\hphantom{-1 \quad 1}}^{C_0}} & \overbrace{\hphantom{0}}^{C_1} & \multicolumn{5}{c}{\overbrace{\hphantom{-1 \quad 1 \quad 1 \quad -1 \quad 1}}^{C_0}} & \overbrace{\hphantom{0}}^{C_1} & \overbrace{\hphantom{-1}}^{C_0}
\end{array} \\
\left(\begin{array}{cccccccccc}
-1 & & & -1 & 1 & & & & & \\
 & 1 & & 1 & -1 & & & & & \\
 & & & & & 1 & & 1 & & -1 \\
 & & & & & -1 & -1 & & & \\
 & & & & & & 1 & -1 & &
\end{array}\right)\left.\begin{array}{c}\\ \\ \\ \\ \\ \end{array}\right\} R_0
\end{array}.$$

Now $Q^\#$ can be seen as a block diagonal matrix consisting of one diagonal block $Q_0^\# = Q^\#[R_0, C_0]$ by $Q^\#[R_0, C_1] = O$. We transform

$$Q_0^\#[R_0, C_0 \cap J^*] = \begin{pmatrix}
-1 & & -1 & 1 & & & \\
 & 1 & 1 & -1 & & & \\
 & & & & 1 & & 1 \\
 & & & & -1 & -1 & \\
 & & & & & 1 & -1
\end{pmatrix}$$

into

$$UQ_0^\#[R_0, C_0 \cap J^*] = \begin{pmatrix}
-1 & & -1 & 1 & & & \\
 & 1 & 1 & -1 & & & \\
 & & & & & & \\
 & & & & -1 & -1 & \\
 & & & & & 1 & -1
\end{pmatrix},$$

where $U = \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & 1 & 1 \\ & & & 1 & \\ & & & & 1 \end{pmatrix}$. Using $U_Q(s) = U$, we modify $A(s)$ to

$$A'(s) = \left(\begin{array}{ccccc|ccccc} -1 & & & -1 & 1 & & & & & \\ & 1 & 1 & 1 & -1 & & & & & \\ \hline & & & & & & & & 1 & -1 \\ & & & & & -1 & -1 & & 1 & \\ & & & & & & 1 & -1 & & \\ \hline R_1 & & & & & -1 & & & & \\ & R_2 & & & & & -1 & & & \\ & & Ls & & & & & -1 & & \\ & & & -1 & & & & & Cs & \\ & & & & & & & & & 1 \end{array}\right),$$

where the third row is different between $A(s)$ and $A'(s)$. The dual solution is updated to $p' = (0, 0, 1, 0, 0, 0, 0, 0, 0, 1)$ and $q' = (0, 0, 1, 0, 0, 0, 0, 0, 1, 1)$. Since the corresponding tight coefficient matrix of $A'(s)$ is nonsingular, we stop the algorithm. The index of the modified DAE remains at two.

Finally, by applying the MS-algorithm to the modified DAE, we obtain an index-1 DAE

$$\begin{cases} -i_1 - i_4 + i_5 = 0, \\ i_2 + i_3 + i_4 - i_5 = 0, \\ v_4 - v_5 = 0, \\ z_4^{[1]} - z_5^{[1]} = 0, \\ -v_1 - v_2 + v_4 = 0, \\ v_2 - v_3 = 0, \\ R_1 i_1 - v_1 = 0, \\ R_2 i_2 - v_2 = 0, \\ L\dot{i}_3 - v_3 = 0, \\ -i_4 + C z_4^{[1]} = 0, \\ v_5 = V(t), \\ z_5^{[1]} = \dot{V}(t), \end{cases}$$

where $z_4^{[1]}$ and $z_5^{[1]}$ are dummy variables corresponding to $\dot{v}_4$ and $\dot{v}_5$, respectively.

7 Application to Nonlinear DAEs

In this section, we discuss the application of our algorithm to nonlinear DAEs. The $\sigma\nu$-method [2], which is implemented in Mathematica [26], adopts a strategy of treating nonlinear or time-varying terms as independent parameters in the Jacobian matrices of DAEs. We first describe the $\sigma\nu$-method briefly.

Consider an index-2 nonlinear DAE

$$\begin{cases} F_1 : \dot{x}_1 + g(x_2) \quad = f_1(t), \\ F_2 : \dot{x}_1 + x_1 + x_3 = f_2(t), \\ F_3 : \dot{x}_1 \qquad\quad + x_3 = f_3(t), \end{cases} \tag{38}$$

where $g\colon \mathbb{R} \to \mathbb{R}$ is a smooth nonlinear function. Their method constructs two kinds of Jacobian matrices JD and JV as follows:

$$\mathrm{JD} = \left(\frac{\partial F_i}{\partial \dot{x}_j}\right)_{i,j} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad \mathrm{JV} = \left(\frac{\partial F_i}{\partial x_j}\right)_{i,j} = \begin{pmatrix} 0 & \mathrm{d}g/\mathrm{d}x_2 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}.$$

If JD is nonsingular, the DAE is index zero from the implicit function theorem. Otherwise, the method performs the Gaussian elimination on JD (and JV simultaneously) to make a row of zeros in JD. Then the method differentiates the corresponding equation, and checks the nonsingularity of JD again. The main feature of the $\sigma\nu$-method is to treat nonlinear or time-varying terms as "independent parameters" to avoid complicated symbolic manipulations. The method works according to the rule that arithmetical operations and the differentiation of independent parameters generate new independent parameters.

The $\sigma\nu$-method may fail due to this rule. For example, let α_1 be an independent parameter representing $\mathrm{d}g/\mathrm{d}x_2$ in JV. By subtracting the first row from the second and third ones, we obtain

$$\mathrm{JD} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \mathrm{JV} = \begin{pmatrix} 0 & \alpha_1 & 0 \\ 0 & \alpha_2 & 1 \\ 0 & \alpha_3 & 1 \end{pmatrix},$$

where $\alpha_2 = 0 - \alpha_1$ and $\alpha_3 = 0 - \alpha_1$ are newly generated parameters by the rule of arithmetical operations. We differentiate the second and third rows. Then JD and JV are

$$\mathrm{JD} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \alpha_2 & 1 \\ 0 & \alpha_3 & 1 \end{pmatrix}, \quad \mathrm{JV} = \begin{pmatrix} 0 & \alpha_1 & 0 \\ 0 & \alpha_4 & 0 \\ 0 & \alpha_5 & 0 \end{pmatrix},$$

where α_4 and α_5 are parameters corresponding to the derivatives of α_2 and α_3, respectively. Although the Jacobi matrix JD is indeed singular due to $\alpha_2 = \alpha_3$, the $\sigma\nu$-method halts at this point as the method regards α_2 and α_3 as independent. This failure originates from the elimination of matrices involving the independent parameter α_1. We have confirmed that the implementation in Mathematica actually fails on this DAE.

Our algorithm is applied to the same DAE (38) as follows. Let

$$A(s) = \begin{pmatrix} s & \alpha & \\ s+1 & & 1 \\ s & & 1 \end{pmatrix},$$

where α is an independent parameter representing $\mathrm{d}g/\mathrm{d}x_2$. As described in Section 2.4, the MS-algorithm is applicable to the nonlinear DAE (38) if $A(s)$ is upper-tight. The tight coefficient matrix corresponding to a dual optimal solution $p = (0, 0, 0)$ and $q = (1, 0, 0)$ is

$$A^{\#} = \begin{pmatrix} 1 & \alpha & \\ 1 & & 1 \\ 1 & & 1 \end{pmatrix},$$

which is singular. Thus we need to modify the matrix. By the same logic as the discussion in Section 6.2, we can regard $A(s)$ as an LM-polynomial matrix $A(s) = \binom{T(s)}{Q(s)}$, where $T(s)$ corresponds to the first row and $Q(s)$ corresponds to the other two ones in $A(s)$. Then our algorithm modifies $A(s)$ to

$$A'(s) = \begin{pmatrix} s & \alpha & \\ 1 & & \\ s & & 1 \end{pmatrix},$$

which is upper-tight (we omit the detail of this modification). Using an optimal solution $p' = (0, 1, 0)$ and $q' = (1, 0, 0)$ of $\mathrm{D}(A')$, the MS-algorithm obtains a purely algebraic equation

$$\begin{cases} z_1^{[1]} + g(x_2) = f_1(t), \\ x_1 \qquad\qquad\; = f_2(t) - f_3(t), \\ z_1^{[1]} \qquad\qquad = \dot{f}_2(t) - \dot{f}_3(t), \\ z_1^{[1]} + x_3 \qquad = f_3(t), \end{cases}$$

where $z_1^{[1]}$ is a dummy variable corresponding to $\dot{x}_1$.

This example shows that our algorithm works for a DAE to which the existing index reduction algorithm cannot be applied. Our algorithm is expected to rarely cause cancellations between nonlinear terms as the algorithm does not perform the row operations involving independent parameters. Therefore, although the application to nonlinear DAEs remains at the stage of heuristic, we consider that the proposed method can be useful for index reduction of nonlinear DAE.

8 Conclusion

In this paper, we have proposed an index reduction algorithm for linear DAEs whose coefficient matrices are mixed matrices. The proposed method detects numerical cancellations between accurate constants, and transforms a DAE into an equivalent DAE to which the MS-algorithm is applicable. Our algorithm uses combinatorial algorithms on graphs and matroids, based on the combinatorial relaxation framework. The algorithm is expected to keep the sparsity of DAEs because the algorithm modifies only rows of accurate constants, and to run much faster in most cases because the algorithm terminates without modifying the DAEs unless the DAEs have unlucky numerical cancellations. We also have developed a faster algorithm for DAEs whose coefficient matrices are dimensionally consistent. Our algorithms can be applied to nonlinear DAEs by regarding nonlinear terms as independent parameters. Numerical experiments on nonlinear DAEs are left for further investigation.

References

[1] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. SIAM, Philadelphia, 1996.

[2] S. Chowdhry, H. Krendl, and A. A. Linninger. Symbolic numeric index analysis algorithm for differential algebraic equations. *Industrial & Engineering Chemistry Research*, 43:3886–3894, 2004.

[3] W. H. Cunningham. Improved bounds for matroid partition and intersection algorithms. *SIAM Journal on Computing*, 15:948–957, 1986.

[4] J. Edmonds. Matroid partition. In G. B. Dantzig and A. F. Veinott, Jr., editors, *Mathematics of the Decision Sciences*, pp. 335–345. AMS, Providence, RI, 1968.

[5] C. W. Gear. Differential-algebraic equation index transformations. *SIAM Journal on Scientific and Statistical Computing*, 9:39–47, 1988.

[6] S. Iwata and M. Takamatsu. Index reduction via unimodular transformations. *METR* 2017-05, University of Tokyo, 2017.

[7] S. Iwata and M. Takamatsu. Computing the maximum degree of minors in mixed polynomial matrices via combinatorial relaxation. *Algorithmica*, 66:346–368, 2013.

[8] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[9] F. Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation (ISSAC 2014)*, pp. 296–303, 2014.

[10] S. E. Mattsson and G. Söderlind. Index reduction in differential-algebraic equations using dummy derivatives. *SIAM Journal on Scientific Computing*, 14:677–692, 1993.

[11] K. Murota. Use of the concept of physical dimensions in the structural approach to systems analysis. *Japan Journal of Applied Mathematics*, 2:471–494, 1985.

[12] K. Murota. Computing puiseux-series solutions to determinantal equations via combinatorial relaxation. *SIAM Journal on Computing*, 19(6):1132–1161, 1990.

[13] K. Murota. Mixed matrices: irreducibility and decomposition. In *Combinatorial and Graph-Theoretical Problems in Linear Algebra*, pp. 39–71. Springer, New York, 1993.

[14] K. Murota. Combinatorial relaxation algorithm for the maximum degree of subdeterminants: Computing Smith-Mcmillan form at infinity and structural indices in Kronecker form. *Applicable Algebra in Engineering, Communication and Computing*, 6:251–273, 1995.

[15] K. Murota. Computing the degree of determinants via combinatorial relaxation. *SIAM Journal on Computing*, 24:765–796, 1995.

[16] K. Murota. On the degree of mixed polynomial matrices. *SIAM Journal on Matrix Analysis and Applications*, 20:196–227, 1998.

[17] K. Murota. *Matrices and Matroids for Systems Analysis.* Springer, Berlin, 2000.

[18] K. Murota and M. Iri. Structural solvability of systems of equations —a mathematical formulation for distinguishing accurate and inaccurate numbers in structural analysis of systems—. *Japan Journal of Applied Mathematics*, 2:247–271, 1985.

[19] K. Murota, M. Iri, and M. Nakamura. Combinatorial canonical form of layered mixed matrices and its application to block-triangularization of systems of linear/nonlinear equations. *SIAM Journal on Algebraic and Discrete Methods*, 8:123–149, 1987.

[20] J. Oxley. *Matroid Theory.* Oxford University Press, Oxford, 2nd ed., 2011.

[21] C. C. Pantelides. The consistent initialization of differential-algebraic systems. *SIAM Journal on Scientific and Statistical Computing*, 9:213–231, 1988.

[22] J. D. Pryce. A simple structural analysis method for DAEs. *BIT Numerical Mathematics*, 41:364–394, 2001.

[23] C. Shi. *Linear Differential-Algebraic Equations of Higher-Order and the Regularity or Singularity of Matrix Polynomials.* PhD thesis, Technische Universität, Berlin, 2004.

[24] G. Tan, N. S. Nedialkov, and J. D. Pryce. Symbolic-numeric methods for improving structural analysis of differential-algebraic equation systems. In J. Bélair, I. A. Frigaard, H. Kunze, R. Makarov, R. Melnik, and R. J. Spiteri, editors, *Mathematical and Computational Approaches in Advancing Modern Science and Engineering*, pp. 763–773. Springer, Cham, 2016.

[25] J. Unger, A. Kröner, and W. Marquardt. Structural analysis of differential-algebraic equation systems — theory and applications. *Computers and Chemical Engineering*, 19:867–882, 1995.

[26] Wolfram Research, Inc. Numerical Solution of Differential-Algebraic Equations — Wolfram Language Documentation. URL: `http://reference.wolfram.com/language/tutorial/NDSolveDAE.html`. (accessed September 27, 2017).