mathematics

MDPI

Article

A New Block Structural Index Reduction Approach for Large-Scale Differential Algebraic Equations

Juan Tang [1,2,*] and Yongsheng Rao [1,2]

[1] School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou 510006, China; rysheng@gzhu.edu.cn

[2] Institute of Computing Science and Technology, Guangzhou University, Guangzhou 510006, China

* Correspondence: tangjn16@gzhu.edu.cn

Received: 27 October 2020; Accepted: 15 November 2020; Published: 18 November 2020

Abstract: A new generation of universal tools and languages for modeling and simulation multi-physical domain applications has emerged and became widely accepted; they generate large-scale systems of differential algebraic equations (DAEs) automatically. Motivated by the characteristics of DAE systems with large dimensions, high index or block structures, we first propose a modified Pantelides' algorithm (MPA) for any high order DAEs based on the Σ matrix, which is similar to Pryce's Σ method. By introducing a vital parameter vector, a modified Pantelides' algorithm with parameters has been presented. It leads to a block Pantelides' algorithm (BPA) naturally which can immediately compute the crucial canonical offsets for whole (coupled) systems with block-triangular form. We illustrate these algorithms by some examples, and preliminary numerical experiments show that the time complexity of BPA can be reduced by at least $O(\ell)$ compared to the MPA, which is mainly consistent with the results of our analysis.

Keywords: differential algebraic equations; index reduction; block triangular forms

1. Introduction

The recent decades encountered a tremendous progress in the field of multi-disciplinary simulation tools for continuous time discrete variable systems. A new generation of universal tools and languages for modeling and simulation multi-physical domain applications, such as Modelica [1], emerged and became widely accepted [2]; they can be found in commercial and academic high quality implementations and allow to generate large-scale systems of differential algebraic equations (DAEs) automatically. The generated DAEs have many interesting characteristics, such as large scale, high index block structures, which are the major motivation of our work in this paper. In the context of numerical calculation of DAEs, it is well known that a direct numerical simulation without index reduction may not be possible or may provide a bad result [3,4]. In brief, higher index DAEs suffer from a rank deficiency in the subset of algebraic equations. The index of a DAE system is a key notion in the theory for measuring the distance from the given system with a singular Jacobian to the corresponding ordinary differential equations with a nonsingular Jacobian. There are various index concepts in the theory of DAEs. The one related to the structural analysis approach is the "structural index", which is defined by (4). For other indices, we refer the interested readers to [5,6]. High-index DAE systems usually need differentiation to reveal all the system's constraints, which are crucial to determine consistent initial conditions. This procedure is the called "index reduction" of DAEs.

Index reduction in the analysis of DAEs numerical solving is an active technique of research. In [7], Campbell and Gear gave a derivative-array method to reduce DAEs , which may not be applicable to large-scale nonlinear systems. Pantelides in [8] constructed a graph-oriented methodology which gives a systematic way to reduce high-index of DAEs with order one to lower index, by selectively adding differentiated forms of the equations already present in the system. In [9], Mattsson-Söderlind used Pantelides' method as a preprocessing step, and then differentiated equations in the DAE system with the aid of the information obtained by Pantelides' method and replaced some derivatives with dummy variables. The dummy method returns a sparse DAEs if the given DAEs is sparse, which can be applied to large-scale DAEs. Pryce in [10] developed structural analysis method (or Σ method) which is proved to compute the same structural index as Pantelides' method and is a straightforward method for analyzing the structure of DAEs of any order. This approach is based on solving an assignment problem, which can be formulated as an integer linear programming problem. The idea was generalized to a class of partial differential algebraic equations by Wu et al. [11]. In [12,13], Pryce and Nediakov et al. generalized the structural analysis method to the DAE systems with coarse or fine block-triangular form (BTF), and showed that the difference between the global offsets of signature matrix Σ and the local offsets of each diagonal sub-block in Σ with fine BTF is a constant. They compute the fine BTF for system's numerical scheme via valid global offset vectors or the local offsets of each separated coarse block. Qin and Tang et al. in [14] generalized the Σ method for large-scale DAE systems. In addition, there are other index reduction methods for different DAEs [15–21]. We focus on structural index reduction method to directly calculate the global canonical offsets of large-scale DAEs with any high order, which are critical for its efficient solution scheme by combing the Pantelides' method with Pryce's Σ-method.

The paper is organized as follows. In the next section, we first make necessary preparations and briefly reviews about Pryce's Σ-method. Section 3 presents a modified Pantelides' method derived from Pantelides' method based on Σ matrix. In section 4, we introduce the block triangular forms (BTF) for large-scale DAEs, firstly. Based on our modified Pantelides' method with parameter, a block Pantelides' algorithm is proposed to find the canonical offsets of the systems of DAEs. We illustrate the performance of these algorithms via numerical experiments in Section 5. In the last section, some necessary conclusions are made.

2. Preliminaries

In this section, we give a brief review about the main steps of Pryce's structural analysis method or Σ-method [10] and some remarks.

We consider n dimension DAEs as follows

$$\mathbf{f} = (f_1, f_2, \ldots, f_n) = \mathbf{0},$$

where

$$f_i = f_i(t, D(x_j(t)))), 1 \leq i, j \leq n,$$

$x_1(t), x_2(t), \cdots, x_n(t)$ are scalar independent functional variable, $D(x_j(t))$ is the derivatives of $x_j(t), t \in \mathbb{R}$.

Step 1. Built the $n \times n$ signature matrix $\Sigma = (\sigma_{ij})$ of the DAEs, where

$$\sigma_{ij} = \begin{cases} \text{highest differential order of } x_j \text{ in } f_i, \text{ if } x_j \text{ appears in } f_i, \\ -\infty, \text{ otherwise.} \end{cases}$$

Step 2. Solve an assignment problem to find a highest value transversal (HVT) T, which is a subset of sparsity pattern S with n finite entries and describes just one element in each row and each column, such that $\sum \sigma_{ij}$ is maximized and finite. The sparsity pattern S of Σ is defined as:

$$S = \text{sparse}(\Sigma) = \{(i,j) : \sigma_{ij} > -\infty\}. \tag{1}$$

This can be formulated as a Linear Programming Problem, the Primal Problem is:

$$\begin{aligned} \max_{\xi} \quad & z = \sum_{(i,j)\in S} \sigma_{ij}\xi_{ij}, \\ \text{s.t.} \quad & \sum_{j:(i,j)\in S} \xi_{ij} = 1 \text{ for each } i, \\ & \sum_{i:(i,j)\in S} \xi_{ij} = 1 \text{ for each } j, \\ & \xi_{ij} \geq 0 \text{ for } (i,j) \in S. \end{aligned} \tag{2}$$

The problem is equivalent to finding a maximum-weight perfect matching in a bipartite graph whose incidence matrix is the signature matrix, and can be solved by Kuhn-Munkres algorithm [22] whose time complexity is $O(n^3)$.

Step 3. Determine the offsets of the problem, which are the vectors $\mathbf{c} = (c_i)_{1\leq i\leq n}, \mathbf{d} = (d_j)_{1\leq j\leq n}$, the smallest such that $d_j - c_i \geq \sigma_{ij}$, for all $1 \leq i \leq n, 1 \leq j \leq n$, and $d_j - c_i = \sigma_{ij}$ when $(i,j) \in T$.

This problem can be formulated as the dual of (2) in the variables $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ and $\mathbf{d} = (d_1, d_2, \ldots, d_n)$. The Dual Problem is defined as follows:

$$\begin{aligned} \min_{\mathbf{c},\mathbf{d}} \quad & z = \sum_j d_j - \sum_i c_i, \\ \text{s.t.} \quad & d_j - c_i \geq \sigma_{ij} \text{ for all } (i,j), \\ & c_i \geq 0 \text{ for all } i. \end{aligned} \tag{3}$$

Step 4. Compute the system Jacobian matrix $\mathbf{J}$, given by

$$\mathbf{J}_{ij} = \begin{cases} \frac{\partial f_i}{\partial((d_j - c_i)\text{th derivative of } x_j)}, & \text{if this derivative is present in } f_i, \\ 0, & \text{otherwise.} \end{cases}$$

Step 5. Choose a consistent point. If $\mathbf{J}$ is nonsingular at that point, then the solution can be computed with Taylor series or numerical homotopy continuation techniques in a neighborhood of that point. And using the canonical offsets $\mathbf{c}, \mathbf{d}$ of Problem (3), the structural index is then defined as:

$$\nu = \max_i c_i + \begin{cases} 0, & \text{for all } d_j > 0, \\ 1, & \text{for some } d_j = 0. \end{cases} \tag{4}$$

In order to determine the crucial canonical offsets for structural analysis in DAEs system using fixed-point iteration algorithm (FPIA) [10], we introduce some necessary definitions, firstly. Define a natural semi-ordering of vectors in $\mathbb{R}^n$, for $\forall\ \mathbf{a}, \mathbf{b}$, $\mathbf{a} \preceq \mathbf{b}$ if $a_i \leq b_i$ for each i, the canon of offsets is in the sense of ordering $\preceq$. Given Σ of DAEs systems and a corresponding transversal T, for $\forall\ \mathbf{c} = (c_i)(\in \mathbb{R}^n)$, we define a mapping

$$\mathcal{D}(\mathbf{c}) = (d_j), \text{ where } d_j = \max_i(\sigma_{ij} + c_i),$$

and for $\forall\ \mathbf{d} = (d_j)(\in \mathbb{R}^n)$, we define a mapping

$$\mathcal{C}_T(\mathbf{d}) = (c_i^*), \text{ where } c_i^* = d_j - \sigma_{i,j},\ (i,j) \in T.$$

Furthermore, we define the composition mapping $\phi_T(\mathbf{c}) = \mathcal{C}_T(\mathcal{D}(\mathbf{c}))$ from $\mathbb{R}^n$ to $\mathbb{R}^n$. Then we introduce fixed-point iteration algorithm below.

Remark 1. *If the used transversal T is a HVT, Algorithm 1 can find the canonical offsets $\boldsymbol{c}^*$ and $\boldsymbol{d}^*$ for DAE systems by $||\boldsymbol{c}^*||_1 + 1$ iterations at most, where $||\boldsymbol{c}^*||_1 = max_i\{c_i^*\}$.*

Remark 2. *If the Σ matrix of given DAE systems in Problem (3) contains some transversal T, then the canonical dual-optimal pair $\boldsymbol{c}^*$ and $\boldsymbol{d}^*$ can be found in time $O(n^3 + ||\boldsymbol{c}^*||_1 \cdot n^2)$ via the fixed-point iteration algorithm.*

Algorithm 1: Fixed-point Iteration Algorithm (FPIA)

Input: Σ is the signature matrix of DAEs
Output: **c** and **d**

```
Set c' ← 0 ;
Set d ← D(c') ;
Set T is a HVT of Σ computed by Kuhn-Munkres algorithm ;
Set c ← C_T(d) ;
while c ≠ c' do
    c' ← c;
    d ← D(c);
    c ← C_T(d);
end
return c,d
```

3. A Modified Pantelides' Method

In this section, we present a modified Pantelides' method to calculate the canonical offsets for given signature matrix of DAEs system with any high order. Some necessary definitions from are given below.

Definition 1. *For $\Sigma = (\sigma_{ij})$ and $\boldsymbol{d} = (d_1, d_2, \cdots, d_n) \in \mathbb{R}^n$, the Leading Derivative Set $\mathcal{L}$ is defined as*

$$\mathcal{L} = LDS(\Sigma, \boldsymbol{d}) = \{(i,j) | \sigma_{ij} = d_j, i = 1, 2, \ldots, n, j = 1, 2, \ldots, n\}, \tag{5}$$

and let $\mathcal{L}_i$ denote the set of j indices in the ith row:

$$\mathcal{L}_i = \{j | (i,j) \in \mathcal{L}\}, \tag{6}$$

If I is a set of i-indices, define

$$\mathcal{L}(I) = \bigcup_{i \in I} \mathcal{L}_i, \tag{7}$$

that is $\mathcal{L}(I)$ represents the total set of leading derivatives that appear in the I set of equations.

Definition 2. ([8]) *The set I is structurally singular (SS) if $\mathcal{L}(I)$ has fewer elements than I:*

$$|\mathcal{L}(I)| < |I|, \tag{8}$$

where $|\cdot|$ is the cardinality of a set, that is if the I-equations have too few leading derivatives. I is minimal structurally singular (MSS) if it contains no proper SS subset.

In order to locate the MSS subsets for a given DAEs system, we review the Algorithm 2 (see [8]) below.

Algorithm 2: SearchMSS(i,$\mathcal{L}$,MARK,MATCH,PATHFOUND)

Input: i, $\mathcal{L}$, MARK, MATCH, PATHFOUND
Output: MARK, MATCH, PATHFOUND
for $j = 1 : n$ **do**
  if $(i, j) \in \mathcal{L}$ *and MATCH(j)=0* **then**
    MARK(j):=1;
    MATCH(j):=i;
    PATHFOUND:=TRUE;
    return [MARK, MATCH, PATHFOUND] ;
  end
end
for $j = 1 : n$ **do**
  if $(i, j) \in \mathcal{L}$ *and MARK(j):=0* **then**
    MARK(j):=1;
    nI:=MATCH(j);
    CALL SearchMSS(nI,$\mathcal{L}$,MARK,MATCH,PATHFOUND);
    if *PATHFOUND:=TRUE* **then**
      MATCH(j):=i;
      return [MARK, MATCH, PATHFOUND];
    end
  end
end
return [MARK, MATCH, PATHFOUND];

If Algorithm 2 returns PATHFOUND=TURE, there exists an augmenting path emanating from the i-th equation of DAEs. Otherwise, the MSS subsets was found. MARK is a vector for the functional variables of DAEs. If $MARK(j) = 1$, this means that the j-th variable has been marked, or it was not unmarked. In addition, MATCH is a matching vector between the equations and the variables of DAEs. MATCH(j) equals 0 which means the j-th variable was not matched. But if MATCH(j) $= i$, this means that the j-th variable has been matched with the i-th equation.

Remark 3. *If Algorithm 2 returns flag PATHFOUND with value FALSE, then the* $Fset = \{i\} \bigcup \{MATCH(j) | j \in Xset\}$ *of equations is MSS with respect to* $Xset = \{j | MARK(j) = 1\}$, *and* $|Fset| = |Vset| + 1$.

Based on Algorithm 2, a complete Algorithm 3 below can be constructed to determine an necessary differentiations of system equations.

Algorithm 3: Modified Pantelides' Algorithm (MPA)

Input: Signature matrix Σ of nonsingular DAEs
Output: Canonical offsets $\mathbf{c}$ and $\mathbf{d}$

Fset $:= \emptyset$, Xset $:= \emptyset$;
$\mathbf{c} := \mathbf{0}$;
$\mathbf{d} := \mathcal{D}(\mathbf{c})$;
$\mathcal{L} :=$ LDS$(\Sigma, \mathbf{d})$;
MACTH $:= \mathbf{0}$;
for $i = 1 : n$ **do**
    PATHFOUND:=FALSE ;
    repeat
        MARK $:= \mathbf{0}$;
        CALL SearchMSS(i,$\mathcal{L}$,MARK,MATCH,PATHFOUND);
        if *PATHFOUND* = *FALSE* **then**
            Xset $:= \{ j \mid \text{MARK}(j) = 1 \}$;
            Fset $:= \{i\} \cup \{ \text{MATCH}(j) \mid j \in \text{Xset}\}$;
            foreach $n_i \in$ *Fset* **do**
                $c_{n_i} := c_{n_i} + 1$ % Update $\mathbf{c}$;
            end
            $\mathbf{d} := \mathcal{D}(\mathbf{c})$ % Update $\mathbf{d}$;
            $\Sigma' := (\sigma'_{ij}) = (\sigma_{ij} + c_i)$;
            $\mathcal{L} :=$ LDS$(\Sigma', \mathbf{d})$ % Update $\mathcal{L}$;
        end
    until *PATHFOUND* = *TURE*;
end
return $\mathbf{c}$ and $\mathbf{d}$;

Lemma 1. ([10]) *For nonsingular DAE systems, Pantelides' algorithm gives the same canonical offsets of the DAEs as fixed-point iteration algorithm.*

Then we are able to give the following theorem.

Theorem 1. *For n dimension nonsingular DAE systems with any high order, the modified Pantelides' algorithm gives the same canonical offsets as the fixed-point iteration algorithm, and the time of the MPA is* $O(n^3)$.

Proof of Theorem 1. Similarly, the proof of the theorem can follow from the Lemma 1 in [10]. □

Remark 4. *From Theorem 1, the modified Pantelides' algorithm not only is able to calculate the canonical offsets for DAEs with any high order, but also finds a HVT* $T = \bigcup_j \{(MATCH(j), j)\}$ *for S(1), which is similar with the Kuhn-Munkres algorithm.*

Example 1. *The motion of a free pendulum in the Cartesian space can be described by a second-order system of DAEs* $\boldsymbol{f} = (f_1, f_2, f_3) = \boldsymbol{0}$,

$$\begin{cases} 0 = f_1 = & x''(t) - \lambda(t)x(t), \\ 0 = f_2 = & y''(t) - \lambda(t)y(t) + g, \\ 0 = f_3 = & x^2(t) + y^2(t) - L^2, \end{cases} \tag{9}$$

where $L, g > 0$ *are constants.*

From the definition of Σ, labeled by equations and dependent variables, sigma matrix is

$$\Sigma = \begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \end{array} \begin{array}{c} \begin{array}{ccc} x & y & \lambda \end{array} \\ \begin{pmatrix} 2 & & 0 \\ & 2 & 0 \\ 0 & 0 & \end{pmatrix} \end{array}, \tag{10}$$

where a blank denotes $-\infty$.

Now, we use Algorithm 3 to compute the canonical offsets of the DAEs and obtain the computational process below.

From Table 1, we can obtain the canonical offsets $\mathbf{c} = (0, 0, 2)$ and $\mathbf{d} = (2, 2, 0)$, and also get a HVT $\{(f_1, x), (f_3, y), (f_2, \lambda)\}$ via MATCH = (1, 3, 2).

Table 1. Process for computing the canonical offsets via Algorithm 3.

i	c	d	$\mathcal{L}$	MARK	MATCH	PATHFOUND	Xset	Fset
1	(0,0,0)	(2,2,0)	{(1,1),(1,3),(2,2),(2,3)}	(1,0,0)	(1,0,0)	TRUE	∅	∅
2	(0,0,0)	(2,2,0)	{(1,1),(1,3),(2,2),(2,3)}	(0,1,0)	(1,2,0)	TRUE	∅	∅
$3^{(1)}$	(0,0,0)	(2,2,0)	{(1,1),(1,3),(2,2),(2,3)}	(0,0,0)	(1,2,0)	FALSE	∅	{3}
$3^{(2)}$	(0,0,1)	(2,2,0)	{(1,1),(1,3),(2,2),(2,3)}	(0,0,0)	(1,2,0)	FALSE	∅	{3}
$3^{(3)}$	(0,0,2)	(2,2,0)	{(1,1),(1,3),(2,2),(2,3),(3,1),(3,2)}	(0,1,1)	(1,3,2)	TRUE	∅	∅

From step 4 in Section 2, the system Jacobian matrix $\mathbf{J}$ is

$$\mathbf{J} = \begin{pmatrix} \frac{\partial f_1}{\partial x''} & 0 & \frac{\partial f_1}{\partial \lambda} \\ 0 & \frac{\partial f_2}{\partial y''} & \frac{\partial f_2}{\partial \lambda} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -x \\ 0 & 1 & -y \\ 2x & 2y & 0 \end{pmatrix}$$

and $\det \mathbf{J} = 2x^2 + 2y^2 = 2L^2 \neq 0$, which means that the modified Pantelides' algorithm succeeds.

4. Structural Index Reduction Methods for DAEs with Block Structure

In this section, assuming that a given DAE system is structurally nonsingular, i.e., the Σ matrix of the system contains at least one transversal, we can get the block triangular forms of signature matrix M by permuting its rows and columns [23,24],

$$M = \begin{pmatrix} M_{1,1} & M_{1,2} & \cdots & M_{1,\ell} \\ & M_{2,2} & \cdots & M_{2,\ell} \\ & & \ddots & \vdots \\ & & & M_{\ell,\ell} \end{pmatrix}, \tag{11}$$

where the elements in the blanks of M are $-\infty$, and the diagonal matrix $M_{i,i}$ is square and irreducible.

The bipartite graph of M or DAE is the undirected graph whose $2n$ vertices are the n rows and n columns, and which has an edge between row i and column j whenever $(i, j) \in S$. We define a structurally nonsingular system of DAE is a coupled system if its graph is connected. In other words, a coupled system does not contains separated subsystems, or the structural analysis can be done in parallel over each subsystem directly and the global canonical offsets of the whole system can be assembled naturally from the local canonical offsets of subsystems.

Next, we propose a block Pantelides' method for the Σ matrix of a coupled system with BTF whose main idea is to use the modified Pantelies' method with parameter mentioned below to process each diagonal matrix in block upper triangulated signature matrix from top to bottom in sequence.

4.1. Modified Pantelides' Method with Parameter

For any given nonnegative parameter $\mathbf{p}$, we present a modified Pantelides' algorithm with parameter (PMPA) just derived from modified Pantelides' algorithm, and obtain the following Lemma from Theorem 1 easily.

Lemma 2. *For a given nonnegative parameter $\boldsymbol{p}$ and a nonsingular DAEs system, the modified pantelides' algorithm with parameter can find the canonical offsets $\boldsymbol{c}$ and $\boldsymbol{d}$ which satisfies with $\boldsymbol{d} \succeq \boldsymbol{p}$. And the time of the PMPA is $O(n^3)$.*

It is noted that if $p_j \leq \max_i \sigma_{ij}$ for all j, then obtain $\mathbf{d} = \max\{\mathcal{D}(\mathbf{0}), \mathbf{p}\} = \mathcal{D}(\mathbf{0})$, which means that the Algorithm 4 is the Algorithm 3.

Example 2. *Consider the DAE system from Example 1, and assume parameter $\boldsymbol{p} = (0, 3, 0)$, then we also get sigma matrix is*

$$\Sigma = \begin{array}{c} \boldsymbol{p} \\ \\ f_1 \\ f_2 \\ f_3 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 3 & 0 \\ x & y & \lambda \end{array} \\ \begin{pmatrix} 2 & & 0 \\ & 2 & 0 \\ 0 & 0 & \end{pmatrix} \end{array}. \tag{12}$$

Similarly, we use Algorithm 4 to compute the canonical offsets of the DAEs and obtain the computational process below.

From Table 2, we can obtain the canonical offsets $\mathbf{c} = (1, 1, 3)$ and $\mathbf{d} = (3, 3, 1)$ which satisfies with $\mathbf{d} = (3, 3, 1) \succeq \mathbf{p} = (0, 3, 0)$.

Algorithm 4: Modified Pantelides' Algorithm with Parameter (PMPA)

Input: Σ is signature matrix of nonsingular DAE systems;
$\mathbf{p}$ is nonnegative parameter vector
Output: Canonical offsets $\mathbf{c}$ and $\mathbf{d}$

```
Fset := ∅, Xset := ∅ ;
c := 0 ;
d := max{D(c), p} ;
L := LDS(Σ, d) ;
MACTH := 0 ;
for i = 1 : n do
    PATHFOUND:=FALSE ;
    repeat
        MARK := 0 ;
        CALL SearchMSS(i,L,MARK,MATCH,PATHFOUND);
        if PATHFOUND = FALSE then
            Xset := { j | MARK(j) = 1 };
            Fset := {i} ∪ { MATCH(j) | j ∈ Xset};
            foreach n_i ∈ Fset do
                c_{n_i} := c_{n_i} + 1      % Update c;
            end
            d := max{D(c), p}               % Update d;
            Σ' := (σ'_ij) = (σ_ij + c_i);
            L := LDS(Σ', d)    % Update L;
        end
    until PATHFOUND = TURE;
end
return c and d;
```

Table 2. Process for computing the canonical offsets via Algorithm 4.

i	$\mathbf{c}$	$\mathbf{d}$	$\mathcal{L}$	MARK	MATCH	PATHFOUND	Xset	Fset
1	$(0,0,0)$	$(2,3,0)$	$\{(1,1),(1,3),(2,3)\}$	(1,0,0)	(1,0,0)	TRUE	$\emptyset$	$\emptyset$
2	$(0,0,0)$	$(2,3,0)$	$\{(1,1),(1,3),(2,3)\}$	(0,0,1)	(1,0,2)	TRUE	$\emptyset$	$\emptyset$
$3^{(1)}$	$(0,0,0)$	$(2,3,0)$	$\{(1,1),(1,3),(2,3)\}$	(0,0,0)	(1,0,2)	FALSE	$\emptyset$	$\{3\}$
$3^{(2)}$	$(0,0,1)$	$(2,3,0)$	$\{(1,1),(1,3),(2,3)\}$	(0,0,0)	(1,0,2)	FALSE	$\emptyset$	$\{3\}$
$3^{(3)}$	$(0,0,2)$	$(2,3,0)$	$\{(1,1),(1,3),(2,3),(3,1)\}$	(1,0,3)	(1,0,2)	FALSE	{1,3}	$\{1,2,3\}$
$3^{(4)}$	$(1,1,3)$	$(3,3,1)$	$\{(1,1),(1,3),(2,2),(2,3),(3,1),(3,2)\}$	(0,1,0)	(1,3,2)	TRUE	$\emptyset$	$\emptyset$

4.2. Block Pantelides' Method for DAEs with BTF

Since a coupled DAE system is structurally nonsingular, we can obtain the signature matrix M with BTF (11), where $\sum_{i=1}^{\ell} n_i = n$ and n_i is the order of M_{ii}.

We give some necessary symbols and definitions from [25] firstly. Assum the parameter vector is $\mathbf{p} = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_\ell)$ with ℓ sections, the offsets are $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_\ell)$ and $\mathbf{d} = (\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_\ell)$, where the dimension of $\mathbf{p}_i$, $\mathbf{c}_i$ and $\mathbf{d}_i$ are n_i for $i = 1, 2, \ldots, \ell$. For any $n \times r$ order matrix B and B' ,n order

vector $\mathbf{q}$, $\bar{\mathbf{q}}$ and $\hat{\mathbf{q}}$, r order vector $\mathbf{w}$, the mapping $B' = \mathbf{RowAdd}(B, \mathbf{q})$ is defined as $B'_{i,j} = B_{i,j} + q_i$, for $i = 1, 2, \ldots, n, j = 1, 2, \ldots, r$; the mapping $\mathbf{w} = \mathbf{ColMax}(B)$ is defined via $w_j = \max_{i \in \{1,2,\ldots,n\}} B_{i,j}$, for $j = 1, 2, \ldots, r$.

Then we are able to propose Block Pantelides' algorithm (BPA) for DAEs with block triangular forms by using PMPA.

Lemma 3. *Assume that the DAE system is a coupled system, then the block Pantelides' algorithm gives the same canonical offsets for the system as the modified Pantelides' algorithm.*

Proof of Lemma 3. Similarly, the proof of the Lemma can follow from the Lemma 3.5 in our former article [25]. □

Moreover, we can obtain the following theorem by Lemma 2 and 3 naturally.

Theorem 2. *Assume a system of DAE is a coupled system with BTF (11), then the canonical offsets $\mathbf{c}^*$ and $\mathbf{d}^*$ of the system can be found in time $O(\sum_{i=1}^{\ell} {n_i}^3)$ via the block Pantelides' algorithm. Furthermore, if $n_i = r$ for each i, i.e., $n = \ell \cdot r$, then the time is $O(\frac{1}{\ell^2} \cdot n^3)$.*

Proof of Theorem 2. Combining Remark 2, Lemma 1 with Lemma 3, block Pantelides' algorithm can obtain the canonical offsets $\mathbf{c}^*$ and $\mathbf{d}^*$ of the system.

Now, we consider the running time of the algorithm. The time of step 4 in algorithm is $O(n_1^3)$ by Lemma 2. For $i = 2 : \ell$, the running time of step 6 is $O((n - \sum_{h=1}^{i-1} n_h) \times n_{i-1})$, the time of step 7 and 8 is $O(\sum_{v=1}^{i-1} n_v \times n_i)$, and the running time of step 9 is $O(n_i^3)$ by Lemma 2. Thus, the elapsed time of the algorithm is

$$O(\sum_{i=1}^{\ell} {n_i}^3 + \sum_{i=2}^{\ell} (n - \sum_{h=1}^{i-1} n_h) \times n_{i-1} + \sum_{i=2}^{\ell} \sum_{v=1}^{i-1} n_v \times n_i) \tag{13}$$

It is noted that

$$(\sum_{i=1}^{\ell} n_i)^2 \leq \ell(\sum_{i=1}^{\ell} n_i^2) \leq \sum_{i=1}^{\ell} {n_i}^3 \tag{14}$$

Then we have

$$\sum_{i=2}^{\ell} (n - \sum_{h=1}^{i-1} n_h) \times n_{i-1} < (\sum_{i=1}^{\ell} n_i)^2 \leq \sum_{i=1}^{\ell} {n_i}^3 \tag{15}$$

and

$$\sum_{i=2}^{\ell} \sum_{v=1}^{i-1} n_v \times n_i < (\sum_{i=1}^{\ell} n_i)^2 \leq \sum_{i=1}^{\ell} {n_i}^3 \tag{16}$$

Therefore, the time complexity of the algorithm is $O(\sum_{i=1}^{\ell} {n_i}^3)$. Moreover, if $n_i = r$ for each i, then the time is $O(\frac{1}{\ell^2} \cdot n^3)$. □

Example 3. *Consider a 6 dimension DAEs system* $\boldsymbol{f} = (f_1, f_2, f_3, f_4, f_5, f_6) = \boldsymbol{0}$ *as follows,*

$$\begin{aligned}
0 = f_1 &= x_1''(t) - x_3(t)x_1(t), \\
0 = f_2 &= x_2''(t) - x_3(t)x_2(t) + g, \\
0 = f_3 &= x_1^2(t) + x_2^2(t) + x_5'(t) - L_1^2, \\
0 = f_4 &= x_4''(t) - x_6(t)x_4(t), \\
0 = f_5 &= x_5''(t) - x_6 x_5(t) + g, \\
0 = f_6 &= x_4^2(t) + x_5^2(t) - L_2^2,
\end{aligned}$$

where $L_1,\ L_2,\ g > 0$ *are constants.*

Firstly, we get the signature matrix Σ with BTF (17), and use Algorithm 5 to compute the canonical offsets $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2)$ and $\mathbf{d} = (\mathbf{d}_1, \mathbf{d}_2)$ of the DAEs, where

$$\Sigma = \begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \left(\begin{array}{cccccc} 2 & & 0 & & & \\ & 2 & 0 & & & \\ 0 & 0 & & & 1 & \\ \hline & & & 2 & & 0 \\ & & & & 2 & 0 \\ & & & 0 & 0 & \end{array}\right) \end{array} = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ & \Sigma_{22} \end{pmatrix}. \tag{17}$$

For $i = 1$, $\mathbf{p}_1 = \mathbf{0}$, then $(\mathbf{c}_1, \mathbf{d}_1)$ = PMPA($\Sigma_{11}, \mathbf{0}$) = MPA(Σ_{11}). From Example 1, obtain $\mathbf{c}_1 = (0,0,2)$ and $\mathbf{d}_1 = (2,2,0)$. For $i = 2$, $\mathbf{p}_2 = max\{ColMax(RowAdd(\Sigma_{12}, \mathbf{c}_1)), \mathbf{0}\} = (0,3,0)$, then $(\mathbf{c}_2, \mathbf{d}_2) = PMPA(\Sigma_{11}, \mathbf{p}_2)$. From Example 2, get $\mathbf{c}_2 = (1,1,3)$ and $\mathbf{d}_2 = (3,3,1)$. Therefore, we obtain whole canonical offsets $\mathbf{c}_2 = (0,0,2,1,1,3)$ and $\mathbf{d}_2 = (2,2,0,3,3,1)$ for the DAEs.

Algorithm 5: Block Pantelides' Algorithm (BPA)

Input: Σ matrix M of a coupled DAE system with BTF (11)

Output: Canonical offsets $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_\ell)$ and $\mathbf{d} = (\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_\ell)$

1 $\mathbf{p} := (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_\ell)$, $\mathbf{p}_j = \mathbf{0}$, $j = 1, 2, \ldots, \ell$;

2 $\mathbf{c} := (\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_\ell)$, $\mathbf{c}_j = \mathbf{0}$, $j = 1, 2, \ldots, \ell$;

3 $\mathbf{d} := (\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_\ell)$, $\mathbf{d}_j = \mathbf{0}$, $j = 1, 2, \ldots, \ell$;

4 $(\mathbf{c}_1, \mathbf{d}_1) \leftarrow$ PMPA($\Sigma_{11}, \mathbf{p}_1$);

5 **for** $i = 2 : \ell$ **do**

6 $\quad [\Sigma_{i-1,i}, \ldots, \Sigma_{i-1,\ell}] \leftarrow$ **RowAdd**$([\Sigma_{i-1,i}, \ldots, \Sigma_{i-1,\ell}], \mathbf{c}_{i-1})$ % Update;

7 $\quad \mathbf{p}_i \leftarrow$ **ColMax**$\left(\begin{pmatrix} \Sigma_{1,i} \\ \vdots \\ \Sigma_{i-1,i} \end{pmatrix}\right)$ % Update;

8 $\quad \mathbf{p}_i \leftarrow \max\{\mathbf{p}_i, \mathbf{0}\}$;

9 $\quad (\mathbf{c}_i, \mathbf{d}_i) \leftarrow$ PMPA($\Sigma_{ii}, \mathbf{p}_i$);

10 **end**

11 **return** **c** and **d**;

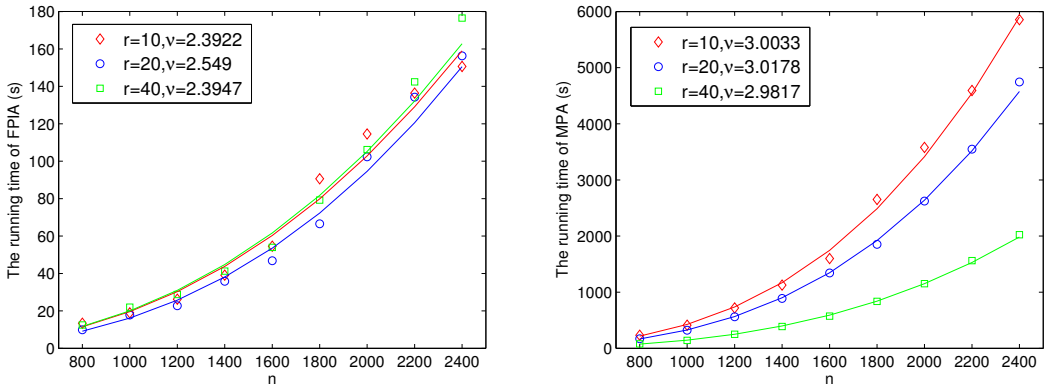
5. Numerical Experimentation

In this section, we do numerical simulation experiments about the running time of structural analysis algorithms for the DAE systems with different scale.

For large-scale or sparse systems produced by multi-domain unified modeling techniques, they always contain several (coupled) subsystems. Considering about the low complexity of block triangulation algorithms [23], we assume that the signature matrices M of coupled systems have been changed into the BTFs (11) and that the order of each diagonal block M_{ii} in $n \times n$ order matrix M is r in our experiments, that is $n = r \cdot \ell$. It is worth noting that the entries of M are the differential orders in DAEs. We can randomly generate the sparse M for coupled systems with BTFs simply and properly as follows:

- for each element in $M_{1,1}$, randomly select an integer from {0, 1, 2, 3} according to its probability from {0.7, 0.1, 0.1, 0.1}, respectively; and $M_{ii} = M_{1,1}, i = 2, 3, \ldots, \ell$;
- for each element in $M_{1,2}$, randomly select an integer from {-1000, 0, 1, 2} according to its probability from {0.9, 0.05, 0.025, 0.025}, respectively; and $M_{i,i+1} = M_{1,2}, i = 2, 3, \ldots, \ell - 1$;
- the rest elements in M are -1000 (means $-\infty$).

All codes are written in Matlab 2016a under Windows 10 system and run on a personal computer with Intel(R) Core(TM) i5-3570 CPU @ 3.40 GHz, 4.00 GB RAM and 64-bit operating system. We do corresponding random trials for FPIA, extended signature matrix method (ESMM [14]), MPA and BPA with $r = \{10, 20, 40\}$ and $n = 800 : 200 : 2400$, and calculate their constants in $\mu \cdot n^{\nu}$ using the standard least-square method. The running times of these algorithms are shown in Figure 1, respectively; some ratios of the running times are given in Figure 2.



(**a**) (**b**)

Figure 1. *Cont.*

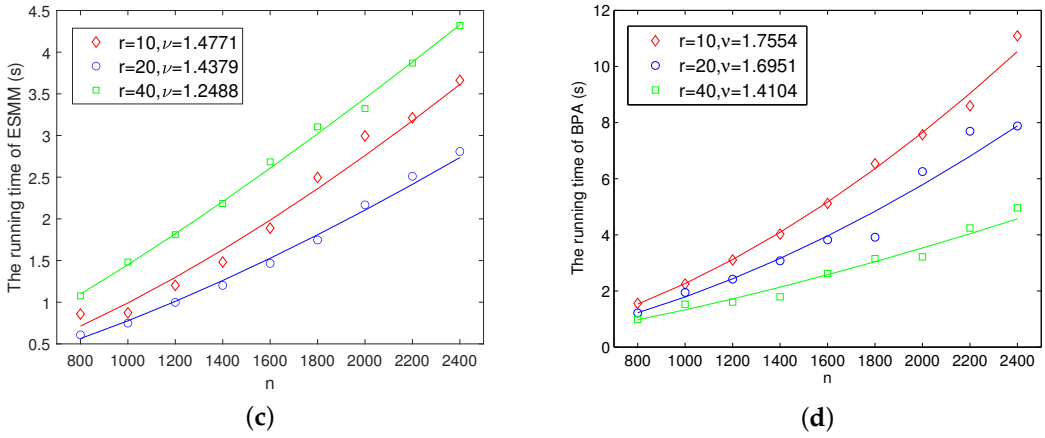


Figure 1. The running times of structural index reduction methods versus n, (**a**) is fixed-point iteration algorithm (FPIA), (**b**) is modified Pantelides' algorithm (MPA), (**c**) is extended signature matrix method (ESMM) and (**d**) is block Pantelides' algorithm (BPA). The constants in the legends are the fitted values ν in μn^{ν} for different r.

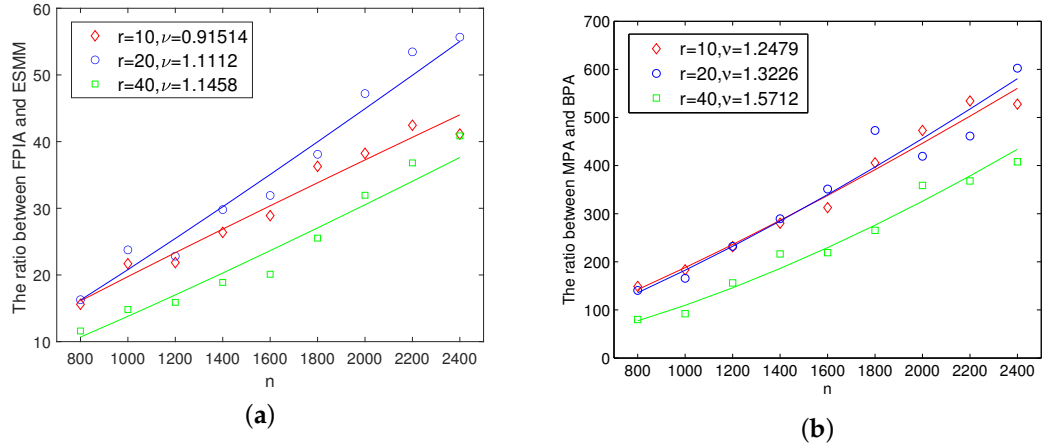


Figure 2. (**a**) The ratio, i.e., $\frac{\text{FPIA's running time}}{\text{ESMM's running time}}$, versus n. (**b**) The ratio, i.e., $\frac{\text{MPA's running time}}{\text{BPA's running time}}$, versus n. The constants in the legends are the fitted values ν in μn^{ν} for different r.

As we can observe in Figure 1, we empirically know that the running times of the BPA for the large-scale DAE systems with different r are between $O(n)$ and $O(n^2)$, FPIA is more like $O(n^{2.5})$, and ESMM is between $O(n)$ and $O(n^{1.5})$. However, MPA is more like $O(n^3)$ which is time consuming because of its recursive operations. It is worth noting that the experimental results of the MPA are consistent with Theorem 1.

Observing from Figure 2, we also know that the ESMM may reduce its running time of large systems for fixed r by nearly $O(n)$ compared to FPIA, while BPA can reduce its running time by at least $O(n)$ (i.e., $O(\ell)$) compared to MPA which is, by and large, consistent with Theorem 2.

In addition, for a fixed n, the running time of the FPIA and ESSM are not influenced basically by the r, while the time of the MPA and BFA decrease with the increase of r. It means that the MPA and BFA may have taken advantage of the intrinsic structure of DAEs.

6. Conclusions

In this article, we first propose a modified Pantelides' method for any high order DAEs based on its Σ matrix, which is similar to Pryce's Σ method and also can find a highest value transversal. By introducing a vital parameter vector, a modified Pantelides' method with parameter has been presented. It leads to a block Pantelides' method naturally which can be applied to immediately calculate the crucial canonical offsets for large-scale (coupled) systems of DAEs with block-triangular form. The main idea of the block Pantelides' method is to use the modified Pantelies' method with parameter to address each diagonal matrix in block upper triangulated signature matrix from top to bottom in sequence. In addition, some examples explain these proposed method clearly, and preliminary numerical experiments show that the time complexity of block Pantelides' algorithm can be reduced by at least $O(\ell)$ compared to the modified Pantelides' algorithm.

As the examples showed in [16,26,27], structural index reduction methods including our methods, which ignore numerical information, may fail on some DAEs due to production of a singular Jacobian. Some researchers proposed different index reduction methods [16,17,19,20] to address this problem for special DAEs which we will consider in the future. Compared with these index reduction methods, our method is able to address a fairly wide class of large-dimension systems of DAEs precisely and efficiently. In the future, the proposed approach will also be applied to exploit some large-scale fractional DAEs [28] or partial DAEs.

Author Contributions: J.T. contributed to supervision, methodology, validation and project administration. Y.R. contributed some computations. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partly supported by the National Key R&D Program of China (Nos. 2018YFB1005100, 2018YFB1005104), and the Science and Technology Program of Guangzhou, China (No. 202002030138).

Acknowledgments: The authors wish to thank the referee for his or her very helpful comments and useful suggestions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Fritzson, P. *Principles of Object-Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*; Wiley-IEEE Press: Piscataway, New Jersey, USA, 2015.
2. Elsheikh, A.; Wiechert, W. The structural index of sensitivity equation systems. *Math. Comput. Model. Dyn. Syst.* **2018**, *24*, 573–592. [CrossRef]
3. Gear, C.W.; Petzold, L.R. ODE methods for the solution of differential/algbraic systems. *SIAM J. Numer. Anal.* **1983**, *21*, 716–728. [CrossRef]
4. Petzold, L.R. Differential/algebraic equations are not ODEs. *SIAM J. Sci. Stat. Comp.* **1982**, *3*, 367–384. [CrossRef]
5. Lamour, R.; Tischendorf, C.; März, R. *Differential-Algebraic Equations: A Projector Based Analysis*; Springer: Berlin/Heidelberg, Germany, 2013.
6. Rheinboldt, W.C. Differential-algebraic systems as differential equations on manifolds. *Math. Comp.* **1984**, *43*, 473–482. [CrossRef]
7. Campbell, S.L.; Gear, C.W. The index of general nonlinear DAEs. *Numer. Math.* **1995**, *72*, 173–196. [CrossRef]
8. Pantelides, C.C. The consistent initialization of differential-algebraic systems. *SIAM J. Sci. Stat. Comput.* **1988**, *9*, 213–231. [CrossRef]
9. Mattsson, S.E.; Söderlind, G. Index reduction in differential-algebraic equations using dummy derivatives. *SIAM J. Sci. Comput.* **1993**, *14*, 677–292. [CrossRef]
10. Pryce, J.D. A simple structural analysis method for DAEs. *BIT* **2001**, *41*, 364–394. [CrossRef]
11. Wu, W.Y.; Reid, G.; Ilie, S. Implicit riquier bases for PDAE and their semi-discretizations. *J. Symb. Comput.* **2009**, *44*, 923–941. [CrossRef]

12. Nedialkov, N.S.; Pryce, J.D.; Tan, G.N. Algorithm 948: DAESA-A matlab tool for structural analysis of DAEs: Software. *ACM Trans. Math. Softw.* **2015**, *41*, 12. [CrossRef]
13. Pryce, J.D.; Nedialkov, N.S.; Tan, G.N. DAESA-A matlab tool for structural analysis of DAEs: Theory. *ACM Trans. Math. Softw.* **2015**, *41*, 9. [CrossRef]
14. Qin, X.L.; Tang, J.; Feng, Y.; Bachmann, B.; Fritzson, P. Efficient index reduction algorithm for large scale systems of differential algebraic equations. *Appl. Math. Comput.* **2016**, *277*, 10–22. [CrossRef]
15. Takamatsu, M.; Iwata, S. Index reduction for differential-algebraic equations by substitution method. *Linear Algebra Appl.* **2008**, *429*, 2268–2277. [CrossRef]
16. Iwata, S.; Oki, T.; Takamatsu, M. Index Reduction for Differential-algebraic Equations with Mixed Matrices. *J. ACM* **2019**, *66*, 35. [CrossRef]
17. Chowdhry, S.; Krendl, H.; Linninger, A.A. Symbolic numeric index analysis algorithm for differential algebraic equations. *Industr. Eng. Chem. Res.* **2004**, *43*, 3886–3894. [CrossRef]
18. Iwata, S.; Takamatsu, M. Index reduction via unimodular transformations. *SIAM J. Matrix Anal. Appl.* **2018**, *39*, 1135–1151. [CrossRef]
19. Oki, T. Improved structural methods for nonlinear differential-algebraic equations via combinatorial relaxation. In *Proceedings of the 44th International Symposium on Symbolic and Algebraic Computation (ISSAC19)*; ACM: New York, NY, USA, 2019.
20. Tan, G.; Nedialkov, N.S.; Pryce, J.D. Conversion methods for improving structural analysis of differential-algebraic equation systems. *BIT Numer. Math.* **2017**, *57*, 845–865. [CrossRef]
21. Qin, X.L.; Yang, L.; Feng, Y.; Bachmann, B.; Fritzson, P. Index reduction of differential algebraic equations by differential Dixon resultant. *Appl. Math. Comput.* **2018**, *328*, 189–202. [CrossRef]
22. Schrijver, A. *Combinatorial Optimization: Polyhedra and Efficiency*; Springer: Berlin/Heidelberg, Germany, 2004.
23. Pothen, A.; Fan, C. Computing the block triangular form of a sparse matrix. *ACM Trans. Math. Softw.* **1990**, *16*, 303–324. [CrossRef]
24. Duff, I.S.; Uçar, B. On the block triangualr form of symmetric matrices. *SIAM Rev.* **2010**, *53*, 455–470. [CrossRef]
25. Tang, J.; Wu, W.Y.; Qin, X.L.; Feng, Y. Structural Analysis Methods for Differential Algebraic Equations via Fixed-Point Iteration. *J. Comput. Theor. Nanosci.* **2016**, *13*, 7705–7711. [CrossRef]
26. Campell, S.L.; Griepentrog, E. Solvability of general differential algebraic equations. *SIAM J. Sci. Comput.* **1995**, *16*, 257–270. [CrossRef]
27. Lamour, R.; März, R. Detecting structures in differential algebraic equations: Computational aspects. *J. Comput. Appl. Math.* **2012**, *236*, 4055–4066. [CrossRef]
28. Tai, Y.P.; Chen, N.; Wang, L.J.; Feng, Z.Y.; Xu, J. A numerical method for a system of fractional differential-algebraic equations based on sliding mode control. *Mathematics* **2020**, *8*, 1134. [CrossRef]

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).