An Extension of Pantelides Algorithm for Consistent Initialization of Differential-Algebraic Equations Using Minimally Singularity

An Extension of Pantelides Algorithm for Consistent Initialization of Differential-Algebraic Equations Using Minimally Singularity

Keisuke Shimako[1†] and Masanobu Koga[2]

[1]Department of Creative Informatics, Kyushu Institute of Technology, Fukuoka, Japan
(Tel: +81-948-29-7500; E-mail: shimako@mk.ces.kyutech.ac.jp)
[2]Department of Intelligent and Control Systems, Kyushu Institute of Technology, Fukuoka, Japan
(Tel: +81-948-29-7500; E-mail: koga@ces.kyutech.ac.jp)

Abstract: In this paper, an algorithm for index reduction of differential algebraic equations (DAE) is proposed. Pantelides algorithm has already been proposed as an algorithm for this purpose. This conventional algorithm has succeeded in reducing the calculation time required for index reduction. However, there exist some DAE systems whose index cannot be reduced correctly with the conventional algorithm. In this paper, we propose an extension of Pantelides algorithm to deal with wider class of DAE systems.

Keywords: differential-algebraic equations, index reduction.

1. INTRODUCTION

Differential Algebraic Equation (DAE) is a system of ordinary differential equations (ODE) and algebraic equations. ODEs often appear to describe phenomena in fields such as science, engineering, and economics using mathematical modeling techniques. Algebraic equations represent constraints that must be satisfied for a phenomenon. Theories and methods of ODE and algebraic equations have been studied for a long time. However, theories regarding DAEs have attracted more recent attention in comparison with them, so that theories regarding DAEs are in a state of flux[1]. DAE systems appear as a problem in mechanical and electrical engineering under constrained conditions. Generally, it is difficult to solve the DAE system. Up until now, this was solved by approximating DAE to ODE, but in recent years, it has been required to solve the DAE without approximation with software tools.

If the problem is well-posed, the DAE system can be analytically differentiated to obtain an ODE for each unknown variable. The minimum number of differentiations required for this is called the "index". In particular, a DAE system with an index of 2 or more is called a high-index DAE system, and it is said that it is difficult to solve because of hidden constraints on the initial conditions.

One way to solve a high index DAE system is to reduce the DAE index. Once we reduce the index of the DAE system to a maximum of 1, we will have a numerically stable solver with an index of 1 and will be able to solve stably. For the above reasons, the high-index DAE system is dealt with by applying index reduction.

MATLAB [2] and Mathematica [3] can solve the DAE system. These tools deal with high-index DAE systems by reducing the index to at most 1.

The tools listed here use Pantelides algorithm [4] to reduce the DAE index. Pantelides algorithm uses the conditions for differentiating the equations of the DAE system and reduces the index of the DAE system until it reaches 1 at most. However, DAE systems whose index can not be reduced under the conditions provided by this algorithm exists. Fig. 1 shows the result when Pantelides algorithm is actually applied in MATLAB to the DAE system where the index cannot be reduced correctly with Pantelides algorithm.

```
>> syms  x(t) y1(t) y2(t)
>> eqs = [diff(x) == x + 2*y1 + 3*y2
       0 == x + y1 + y2 + 1
       0 == 2*x + y1 + y2];
>> vars = [x(t), y1(t), y2(t)];
>> [newEqs, newVars] = reduceDAEIndex(eqs, vars);
警告: Index of reduced DAEs is larger than 1.
> In symengine
  In mupadengine/evalin_internal (line 104)
  In mupadengine/feval_internal (line 167)
  In sym/reduceDAEIndex (line 98)
```

Fig. 1 Failure in MATLAB.

In this example, reduceDAEIndex function of MATLAB[2] is applied to the DAE system

$$\begin{cases} \dot{x} = x + 2y_1 + 3y_2 \cdots f_1 \\ 0 = x + y_1 + y_2 + 1 \cdots f_2 \\ 0 = 2x + y_1 + y_2 \cdots f_3 \end{cases} \tag{1}$$

, where $x(t), y_1(t), y_2(t)$ are system state variables. reduceDAEIndex retains the original equations and variables and generates new variables and equations. [newEqs, newVars] = reduceDAEIndex(eqs, vars) converts a high-index DAE system eqs to an equivalent system newEqs with differential index 1. Pantelides algorithm[4] is used for this process. After the conversion, reduceDAEIndex calls isLowIndexDAE to check the differential index of the new system. isLowIndexDAE function is a function to determine whether the index of DAE system is 1 or less or 2 or more. If the index of newEqs is 2 or more, reduceDAEIndex issues a warning. Since the reduceDAEIndex function was called and a warning was issued, it can be seen that the index of DAE system can not be correctly reduced.

Σ method[5] as well as Pantelides algorithm have been proposed as methods for index reduction. Both meth-

† Keisuke Shimako is the presenter of this paper.

ods have DAE systems that cannot be dealt with. Reference [6] proposed the method to deal with wider class of DAE with Σ method. On the other hand, since tools such as MATLAB use Pantelides algorithm, we consider the method to deal with wider class of DAE with Pantelides algorithm in this paper.

In this paper, we propose an extension of Pantelides Algorithm to deal with wider class of DAE systems. This paper is organized as follows. In Section 2, formulates the problem to be solved. In Section 3, introduces the conventional algorithm for lowering the index of high-index DAE. In Section 4, we propose a new algorithm. In Section 5, the proposed algorithm is evaluated. Finally in Section 6 we conclude the paper.

2. PROBLEM FORMULATION

The problem to be solved in this paper is formulated as follows.

Problem: The DAE systems considered here are of the general form

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

where $x, \dot{x} \in R^n, y \in R^m, f : G \subseteq R^n \times R^n \times R^m \times R \to R^{n+m}$.

Variables appearing in equations are distinguished by x and y. x is a variable where $\dot{x}$ exists in the DAE system to be considered. y is a variable where $\dot{y}$ does not exist in the DAE system to be considered.

In Pantelides algorithm, (2) is represented as

$$f(z, x, t) = 0, \tag{3}$$

using $z = (\dot{x}, y)$ as the new variable that appears when differentiated.

Let the subset of interest be

$$\bar{f}(\bar{z}, \bar{x}, t) = 0. \tag{4}$$

At this time, if there is an $\bar{f}$ in the DAE system to be considered in which the matrix $\bar{f}_{\bar{z}}$ obtained by partially differentiating $\bar{f}$ by $\bar{z}$ has a vector that is not linearly independent, Pantelides algorithm cannot correctly perform index reduction.

From the above, the problem dealt with in this paper is the DAE system represented by (2), which has f_z containing a vector that is not linearly independent.

3. CONVENTIONAL ALGORITHM

In this section, the conventional method is introduced. Pantelides algorithm is mainly used as an index reduction algorithm, but was originally proposed to find consistent initial conditions for DAE systems. When the initial conditions of the DAE system can be obtained, the index is 1 at most, so Pantelides algorithm is used to reduce the index. Since this study extends the Panteliides algorithm, we will discuss not to reduce the index, but to determine the initial conditions of the DAE system. When the initial conditions for the DAE system can be determined, the index for the DAE system can be reduced.

3.1. Subset to be differentiated

When considering the initial conditions of a DAE system, the initial conditions must not only satisfy the original equations, but also satisfy the hidden constraints obtained by differentiation. However, differentiating equations blindly is not efficient because some equations show hidden constraints by differentiation and others do not. Therefore, Pantelides algorithm establishes its own conditions for differentiating the equations, and differentiates only the equations that require differentiation to obtain the initial conditions of the DAE system.

Whether differentiating an equation reveals hidden constraints on the initial conditions is determined not only by that equation, but also by a subset $\bar{f}$ of the equation.

Let $\bar{f}$ be composed of k equations with $\bar{x} \in R^q$ and $\bar{z} \in R^\ell$. Let the row rank of $\bar{f}_{\bar{z}}$ is r. Since $\bar{f}_{\bar{z}}$ is a matrix of k rows and ℓ columns, it holds the relation

$$r \leq \min(\ell, k) \tag{5}$$

.

Based on the above conditions, we consider which subset of equations should be differentiated and the hidden constraints for determining the initial conditions of the DAE system will appear. Assuming that $(\bar{f}_{\bar{z}} : \bar{f}_{\bar{x}})$ has full row rank, k, and that (4) is differentiable, we differentiate (4) with respect to time to obtain:

$$\bar{f}_{\bar{z}}\dot{\bar{z}} + \bar{f}_{\bar{x}}\dot{\bar{x}} + \bar{f}_t = 0. \tag{6}$$

(6) can be transformed into

$$\left(\bar{f}_{\bar{z}} : \bar{f}_{\bar{x}}\right) \begin{pmatrix} \dot{\bar{z}} \\ \dot{\bar{x}} \end{pmatrix} = -\bar{f}_t. \tag{7}$$

With the relationship of (5), (6) can be transformed into

$$\begin{pmatrix} A & B \\ O & C \end{pmatrix} \begin{pmatrix} \dot{\bar{z}} \\ \dot{\bar{x}} \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix} \tag{8}$$

by using the elementary row operations. (8) gives

$$A\dot{\bar{z}} + B\dot{\bar{x}} = a \tag{9}$$

and

$$C\dot{\bar{x}} = b \tag{10}$$

where $A \in R^r \times R^\ell, B \in R^r \times R^q, C \in R^{k-r} \times R^q$ are matrices and $a \in R^r, b \in R^{k-r}$are vectors, all functions of $(\bar{z}, \bar{x}, t)$in general.

Differentiation has introcduced the new variables $\dot{\bar{z}}$ which do not occcur in the original system (2); it has also created the new equations (9) and (10). we note that, because of (5), ℓ is always at least as large as r and one can therefore partition vector $\bar{z}$ into $\bar{z}_1 \in R^r$ and $\bar{z}_2 \in R^{\ell-r}$, and matrix A into $A_1 \in R^r \times R^r$ and $A_2 \in R^r \times R^{\ell-r}$ such that A_1 is nonsingular. Solving the resulting equations, one obtains from (9) an explicit expression for $\dot{\bar{z}}_1$:

$$\dot{\bar{z}}_1 = A_1^{-1}a - A_1^{-1}(B\dot{\bar{x}} + A_2\dot{\bar{z}}_2) \tag{11}$$

Now for any set of values $(\bar{x}, \dot{\bar{x}}, \bar{y})$, one can always find values for the new variables $\dot{\bar{z}}$ such that (9) is satisfied. No

useful information concerning the original variable set is contained in (9).

However, since by assumption rank$(\bar{f}_{\bar{z}} : \bar{f}_{\bar{x}}) = k$, matrix C in (10) has full row rank, $(k - r)$. Thus, (10) constitutes $(k - r)$ new equations which the original variable set must satisfy together with the original set (2).

Therefore, the equations that must be satisfied by a set of consistent initial conditions can be generated by locating all subset (4) of k equations such that

$$r < k \tag{12}$$

3.2. Conventional Method

Next, we show what conditions Pantelides algorithm uses to differentiate the equations. Pantelides algorithm uses

$$\ell < k \tag{13}$$

as a condition for differentiating the equation. A subset of equations that satisfies condition (13) is called *structurally singular* with respect to the variable subset $\bar{z}$. A structurally singular subset is called *minimally structurally singular* (MSS) if none of its proper subsets is structurally singular. Pantelides algorithm differentiates MSS. Then, when (2) is given to the conventional algorithm,

$$\begin{aligned} f(x, \dot{x}, y, t) &= 0 \\ h_i(x_i, \dot{x_i}) &= 0 \end{aligned} \tag{14}$$

is provided. Here each h_i is a new equation relating x_i to $\dot{x_i}$.

The purpose of using such unique conditions to differentiate the equations is to speed up the processing. Assuming a DAE system consists of $(n + m)$ equations, there are $(2^{n+m} - 1)$ non-empty subsets. If we try to find a subset to be differentiated by the condition (12), in the worst case we will have to check the rank of the Jacobi matrix $(2^{n+m} - 1)$ times, and if $(n + m)$ increases, the algorithm will take quite long time. On the other hand, in the case of (13), the subset to be differentiated can be searched only by considering the relationship between the numbers, so that even if $(n + m)$ increases, the calculation time does not increase so much.

Here, because of (5), the condition (13) is a sufficient condition for being (12). Therefore, if (12) but (13), the algorithm terminates without differentiating the equation to be differentiated. Such problems cannot be dealt with Pantelides algorithm.

As described above, the case that the equation to be differentiated is not differentiated occurs when $\bar{f}_{\bar{z}}$ includes a vector that is not linearly independent. Therefore, in the algorithm proposed in this study, (12) is added as a condition for differentiating the equation, and the subset that becomes (12) is defined as a *singular subset*. In addition, similar to Pantelides algorithm, a singular subset is called *minimally singular* (MS) if none of its proper subsets is singular. The proposed algorithm differentiates MS.

4. PROPOSED ALGORITHM

A flowchart of the proposed algorithm is shown in Fig.2.

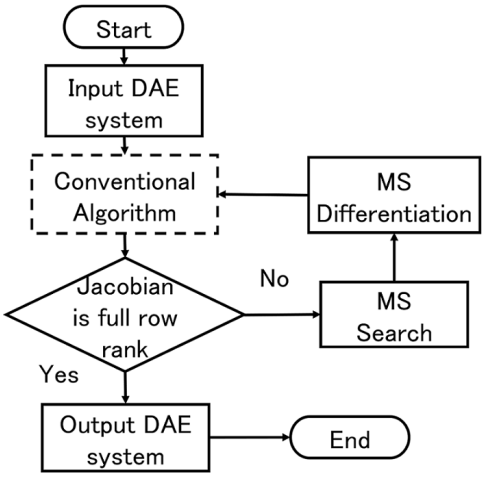

Fig. 2 Flowchart of the proposed algorithm.

The dotted line in Figure 2 is the flowchart of the conventional algorithm, and the solid line is the part added by the proposed algorithm.

The proposed algorithm are described in detail below. First, to the given system apply the conventional algorithm. Next, we use Jacobian matrix to determine whether the initial conditions of the DAE system have been obtained correctly. The initial conditions of the DAE system are not determined correctly when the Jacobian matrix contains vectors that are not linearly independent. In this case, the conditional branch block determines " No ", differentiates the subset that requires differentiation, and reapplies the conventional algorithm. If the Jacobian matrix does not include any vector that is not linearly independent, it is determined as "Yes" and the algorithm ends. This is repeated until the initial conditions of the DAE system can be correctly determined.

The parts added in this study can be divided into
"Checking the Rank of the Jacobian Matrix",
"MS Search" and "MS Differentiation". Each of them is described in detail below.

4.1. Conventional Algorithm

Before searching for the MS, apply Pantelides algorithm to the DAE system once. The reason for doing this is to reduce the number of MSs to look for. Once Pantelides algorithm is applied to the DAE system, the subset that should be differentiated and does not contain a vector that is not linearly independent in $\bar{f}_{\bar{z}}$ is differentiated by Pantelides algorithm. Therefore, only the subset in which $\bar{f}_{\bar{z}}$ contains vectors that are not linearly independent ends without differentiation. From the above, when performing MS search, if Pantelides algorithm is first applied, it is only necessary to search for $\bar{f}_{\bar{z}}$ that includes a vector that is not linearly independent. Pantelides algorithm is faster than the proposed algorithm in determining which equations to differentiate. Therefore, the conven-

tional algorithm may be used in the proposed one in order to reduce the execution time of the entire algorithm.

4.2. Checking rank of jacobian matrix

After applying the conventional algorithm, determine whether the initial condition can be correctly calculated for the equation (14) output from the conventional algorithm. To investigate this, we use the list B, which is also used in the conventional algorithm. List B is defined as:

$$\begin{aligned} B(i) = \ell : \textbf{if} \quad & f_\ell = \dot{f}_i \\ 0 : \text{otherwise.} \end{aligned} \tag{15}$$

$B(i) = 0$ if the i-th equation is not differentiated. Let

$$\hat{f}(x, \dot{x}, y, t) = 0 \tag{16}$$

be a system consisting of equations that are $B(i) = 0$. There are always $(n+m)$ equations for which $B(i) = 0$. So $\hat{f}$ always consists of $(n+m)$ equations.

Here, by considering $\hat{f}$ as one subset and Let $\hat{r}$ be the rank of the Jacobian matrix of $\hat{f}$. From the description of section 3.1, checking whether it holds that

$$\hat{r} < n + m, \tag{17}$$

it is possible to determine whether there is a hidden constraint for determining the initial condition. If (17) is true, differentiation is still required, so MS Search is performed, otherwise the algorithm is terminated.

The loop from "Conventional Algorithm" to "MS Differentiation" in Fig.2 is repeated until the initial conditions can be obtained correctly. When the block of "Check Jacobian Matrix" in the figure does not require any further differentiation, it outputs all equations.

Conventional algorithms do not consider the derivative of the equation, which is $B \neq 0$. Therefore, we can reduce the search time by defining $\hat{f}$ and looking only at it.

4.3. MS Search

MS Search is performed on the subset that can be generated from $\hat{f}$. The MS Search is realized by examining the rank r of the matrix $\bar{f}_{\bar{z}}$ in descending order of k of all possible $\bar{f}$. Equations in the subset where $r < k$ are specified as equations to be differentiated.

At this time, the subset containing the equation specified as the equation to be differentiated once is excluded from MS search. In this way, MS could be searched as a result.

In addition, from the description later in section 3.1, there are only $(k - r)$ subsets that, when differentiated, show hidden conditions for determining initial conditions. Therefore, the MS search ends when $(k - r)$ differentiating subsets are found. In this way, the number of $\bar{f}_{\bar{z}}$ lookups can be reduced as much as possible.

4.4. MS Differentiation

MS differentiates according to the procedure from (6) to (10). Since there is an assumption that MS is (12), one or more (10) is always derived by differentiating in this way. Since (10) does not include z, this will always be a subset with the condition (13) and will be differentiated at the next time conventional algorithm is applied. However, if we don't follow these steps and just make a distinction, we may leave a subset that is not (13) but (12). It can be said that such a differentiation facilitates the analysis when the conventional algorithm is applied next, in that the expression of (10) can be explicitly derived.

Differentiating the equation updates list B and $\hat{f}$. Next, we try to find the initial conditions using the conventional algorithm. At this time, the conventional algorithm is applied to $\hat{f}$. We may be worried that some problems will occur if we do not consider the equation before differentiation. However, even when differentiation occurs in the conventional algorithm, the equation before differentiation is not considered in obtaining the initial condition. Therefore, applying the conventional algorithm to only $\hat{f}$ does not cause any problem.

5. EXAMPLE

5.1. Example

This section describes an example and the execution time.

(1) is used as an example of a DAE system in which the initial conditions cannot be obtained correctly with Pantelides algorithm.

If subset $\bar{f}$ of the system equations is taken to consist of the last two equation $(k = 2)$, then with $\bar{z} = (y_1, y_2)^T$ we have

$$\bar{f}_{\bar{z}} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \tag{18}$$

i.e.,rank$\left(\bar{f}_{\bar{z}}\right) = 1 < 2$(see (12)).

Clearly this subset should be differentiated because of condition (12).

However, since the number ℓ of $\bar{z}$ is also 2, the condition (13) for differentiating the equation used in Pantelides algorithm is not satisfied. Therefore, the algorithm terminates without detecting all the equation subsets which should to be differentiated.

5.2. How the algorithm works

An example of the operation when the algorithm proposed in this study is applied to (1). First, apply Pantelides algorithm. At this time, no differentiation occurs at this stage because (1) has no subset that is (13).

Next, it is determined using the condition $r < k$ whether or not the initial conditions have been correctly obtained for the equation system output from Pantelides algorithm.

At this time, it becomes

$$f_z = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \tag{19}$$

The rank r of f_z is 2, and the number k of equations is 3. Because of the relationship of $r < k$, the algorithm does not end as it is, but starts searching for a subset to be differentiated.

Next, the MS search is started. At this point, list B is

$$B = (0, 0, 0) \tag{20}$$

Therefore, $\hat{f}$ is composed of f_1, f_2, f_3.

All possible subsets in this example are

$$\begin{aligned}\bar{f} = \{&\{f_1\}, \{f_2\}, \{f_3\}, \{f_1, f_2\}, \{f_1, f_3\}, \\ &\{f_2, f_3\}, \{f_1, f_2, f_3\}\}.\end{aligned} \tag{21}$$

Sorting the subsets in ascending order of k. Examine $\bar{f}_{\bar{z}}$ in this order to find the subset to differentiate. At this time, since $(k - r) = 1$, the search ends when a subset to be differentiated is found.

In this case, since the subset consisting of f_2 and f_3 has the relationship of $r < k$, it is determined that this subset is the subset to be differentiated. At this point, the search for the MS ends.

Next, the MS is differentiated. The MS that differentiates is

$$\begin{aligned} &0 = x + y_1 + y_2 + 1 \cdots f_2 \\ &0 = 2x + y_1 + y_2 \cdots f_3 \end{aligned} \tag{22}$$

Differentiating (22) gives

$$\begin{aligned} &0 = \dot{x} + \dot{y_1} + \dot{y_2} \\ &0 = 2\dot{x} + \dot{y_1} + \dot{y_2} \end{aligned} \tag{23}$$

When (23) is transformed, it becomes

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} \dot{y_1} \\ \dot{y_2} \\ \dot{x} \end{pmatrix} = 0 \tag{24}$$

When the Jacobian matrix of (24) is row-basically transformed, it becomes

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \dot{y_1} \\ \dot{y_2} \\ \dot{x} \end{pmatrix} = 0 \tag{25}$$

From equation (25), equation

$$\begin{aligned} &\dot{y_1} + \dot{y_2} + \dot{x} = 0 \cdots f_4 \\ &\dot{x} = 0 \cdots f_5 \end{aligned} \tag{26}$$

can be obtained. Here, f_5 was obtained. This formula corresponds to formula (10) and is a hidden constraint. This formula is a constraint that was not taken into account by the conventional algorithms.

Then list B is updated to be

$$B = (0, 4, 5, 0, 0) \tag{27}$$

Therefore, $\hat{f}$ is composed of f_1, f_4, f_5. Applying the conventional algorithm to this equation again, the following equation is derived.

$$\begin{aligned} &\ddot{x} = \dot{x} + 2\dot{y_1} + 3\dot{y_2} \\ &\dot{x} + \dot{y_1} + \dot{y_2} = 0 \\ &\ddot{x} = 0 \end{aligned} \tag{28}$$

At this time, it becomes

$$f_z = \begin{pmatrix} 1 & -2 & -3 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix} \tag{29}$$

The rank r of f_z is 3, and the number k of equations is 3.

Therefore, it is determined that no further differentiation is necessary, and the algorithm ends.

5.3. Execution Result

We implemented the proposed algorithm as some functions on Maxima[7]. Fig. 3 shows the execution result on Maxima when they are applied to equation (1).

```
(%i18) F: ['diff(x,t) = x + 2*y1 + 3*y2,
        0 = x + y1 + y2 + 1,
        0 = 2*x + y1 + y2];
         dx
(%o18)  [-- = 3 y2 + 2 y1 + x, 0 = y2 + y1 + x + 1, 0 = y2 + y1 + 2 x]
         dt
(%i19) DAEInitialization(F);
                                                             dx
(%o19) [y2 + y1 + x + 1 = 0, y2 + y1 + 2 x = 0, 3 y2 + 2 y1 - -- + x = 0,
                                                             dt
                                                        2               2
           dx        dy2    dy1    dx          dy2      dy1    d x    dx        d x
           -- = 0,   --- + --- + -- = 0, 3 --- + 2 --- - --- + -- = 0, --- = 0]
           dt        dt     dt     dt          dt       dt      2     dt         2
                                                               dt               dt
```

Fig. 3 Execution example in Maxima

(%i 18) defines (1) as F. "DAEInitialization()" is the function for applying the proposed algorithm to equations. (%i 19) applies the proposed algorithm to F. In (%o 19), the initial conditions of the DAE system obtained by the proposed algorithm are outputed.

$$\begin{cases} y_2 + y_1 + x + 1 = 0 \\ y_2 + y_1 + 2x = 0 \\ 3y_2 + 2y_3 - \dot{x} + x = 0 \\ \dot{x} = 0 \\ \dot{y}_2 + \dot{y}_1 + \dot{x} = 0 \\ 3\dot{y}_2 + 2\dot{y}_1 - \ddot{x} + \dot{x} = 0 \\ \ddot{x} = 0 \end{cases} \tag{30}$$

When the initial condition of (1) is calculated by hand, it becomes

$$\begin{cases} \dot{x} = x + 2y_1 + 3y_2 \\ 0 = x + y_1 + y_2 + 1 \\ 0 = 2x + y_1 + y_2 \\ \dot{y}_1 + \dot{y}_2 = -\dot{x} \\ \dot{x} = 0 \\ 2\dot{y}_1 + 3\dot{y}_2 = \ddot{x} - \dot{x} \\ \ddot{x} = 0 \end{cases} \tag{31}$$

Since (30) is identical with the result by hand, it can be said that the initial conditions can be obtained correctly.

5.4. Execution time

We consider the execution time of the proposed algorithm from two viewpoints. First, we consider the time required to determine whether the equations output from Pantelides algorithm should still be differentiated. Second, we consider the effect of the number of equations on execution time.

Table 1 shows the execution environment.

Table 1 execution environment

Processor	Intel core i5
Memory	32GB
OS	Windows 10 64-bit
Maxima	Maxima 5.43.2

The execution time is measured by Maxima's "time" function[7]. This is a function that returns the time, in seconds, used to calculate an output. The time returned is a Maxima estimate of the internal computation time,

not the elapsed time. In the verification of the execution time of this study, we could do it in a very short time compared to seconds, so we measured the time taken to execute 1000 times. Therefore, in this paper, the average of 1000 executions is defined as one execution time.

5.4.1. Checking rank of Jacobian

we consider the time required to determine whether the equations output from Pantelides algorithm should still be differentiated. This can be measured by preparing an example that conventional algorithm can deal with and examining the difference between the execution time of conventional algorithm and the execution time of the proposed algorithm.

We use

$$\begin{cases} \dot{x}_1 = x + 2y_1 + 3y_2 \\ 0 = x + y_1 + y_2 + 1 \\ 0 = 2x + 2y_1 + y_2 \end{cases} \tag{32}$$

as a problem that conventional algorithm can deal with. Table 2 shows the average run time for 10 runs.

Table 2 Execution time

Algorithm	Execution time[ms]
Conventional algorithm	1.241
Proposed algorithm	1.328

The difference in execution times is 87 milliseconds, so in this example we can see that it takes less than 10% extratime to determine if more differentiation is needed.

5.4.2. Number of equations

The effect of the number of equations in the DAE system on the execution time is discussed. We prepare some examples that conventional algorithm cannot deal with and have different numbers of equations, and compare their execution times to discuss the effects.

(1) is used as an example with three equations.

$$\begin{cases} \dot{x} = x + 2y_1 + 3y_2 + 4y_3 \\ 0 = x + y_1 + y_2 + y_3 \\ 0 = 3x + y_1 + y_2 + 2y_3 \\ 0 = 5x + y_1 + y_2 - y_3 \end{cases} \tag{33}$$

is used as an example with four equations.

$$\begin{cases} \dot{x} = x + 2y_1 + 3y_2 + 4y_3 + 5y_4 \\ 0 = 2x + y_1 + y_2 + y_3 + y_4 + 1 \\ 0 = 3x + y_1 + 2y_2 + 2y_3 + 2y_4 \\ 0 = 5x + y_1 + 2y_2 + 3y_3 + 3y_4 \\ 0 = 7x + 2y_1 + 2y_2 + y_3 + y_4 \end{cases} \tag{34}$$

is used as an example with five equations.

Table 3 shows the average execution time for 10 runs.

Table 3 Execution time

DAE	Number of equation	Execution time[ms]
(1)	3	4.945
(33)	4	9.364
(34)	5	17.84

When the number of equations increased by one, the execution time was approximately doubled.

When a DAE system that conventional algorithm cannot deal with is dealt with, assuming that the number of equations in the DAE system is $(n+m)$, in the search part of MS, examine the Jacobi matrix up to $2^{n+m} - 1$ times. Therefore, the calculation time is $O(2^N)$, and when the number of equations increases by one, the execution time is expected to be approximately doubled.

6. CONCLUSION

An algorithm for index reduction of high-index DAE system has been proposed and evaluated. The proposed algorithm can reduce the index of DAE system, which conventional algorithm cannot be applied to. The example shows that it is possible to perform DAE-based index depreciation, which was not possible with conventional algorithm. As for the execution time, for a DAE system that can perform index reduction with conventional algorithm, it can be executed in almost the same time as conventional algorithm. For a DAE system that cannot be index-decreased by conventional algorithm, the calculation time increases as the number of formulas increases.

Finally, this research deals with the same problem formulation as Pantelides algorithm, so it may be applicable to other studies based on Pantelides algorithm. For example, reference [8] reports that the algorithm has been modified to allow it to be applied to the delayed differential algebraic equations. The proposed algorithm may be applicable to such research.

REFERENCES

[1] Uri.M.Ascher, Linda.R.Petzold "Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations", *SIAM*, 2006

[2] The MathWorks Inc, Simulink, http://jp.mathworks.com/products/simulink/.

[3] WOLFRAM, Wolfram, http://www.wolfram.com.

[4] Constantinos C. Pantelides, "The Consistent Initialization Of Diferential Systems", *SIAM J. SCI STAT COMPUT*, Vol.9, No.2, 1988

[5] J. D. Pryce, "A simple structural analysis method for DAEs, *BIT Numer. Math*, Vol.41, No.2, pp. 364-394, 2001

[6] G.Tan, N. S. Nedialkov, and J. D. Pryce, "Conversion methods for improving structural analysis of differential-algebraic equation systems", *BIT Numerical Mathematics*, 57(3):845– 865, 2017.

[7] Sourceforge, Maxima, a Computer Algebra System, http://maxima.sourceforge.net/.

[8] I.Ahrens, B.Unger, "The Pantelides algorithm for delay differential-algebraic equations", *arXiv:1908.01514*, 2020