The Pantelides algorithm for delay differential-algebraic equations

Ines Ahrens*†   Benjamin Unger*‡

January 8, 2020

Abstract

We present a graph-theoretical approach that can detect which equations of a delay differential-algebraic equation (DDAE) need to be differentiated or shifted to construct a solution of the DDAE. Our approach exploits the observation that differentiation and shifting are very similar from a structural point of view, which allows us to generalize the Pantelides algorithm for differential-algebraic equations to the DDAE setting. The primary tool for the extension is the introduction of equivalence classes in the graph of the DDAE, which also allows us to derive a necessary and sufficient criterion for the termination of the new algorithm.

Keywords: delay differential-algebraic equation, structural analysis, Pantelides algorithm

AMS(MOS) subject classification: 34A09, 34K32, 65L80

1   Introduction

We study *delay differential-algebraic equations* (DDAEs) of the form

$$F(t, x(t), \dot{x}(t), x(t - \tau)) = 0 \tag{1.1}$$

on the time interval $[0, T) = [0, T]$ with $\tau > 0$. The DDAE (1.1) is equipped with an (functional) initial condition of the form

$$x(t) = \varphi(t) \qquad \text{for } t \in [-\tau, 0].$$

Differential equations with delay arise in various applications, where the current rate of change does not only depend on the current state but also on past information, for instance, population dynamics, infection disease models and laser dynamics. We refer to [8] and the references therein. Besides, if a dynamical system is controlled based on the current state of the system – a so-called feedback control law – then such a controller often requires some time to measure the current state and to compute the control action, which again results in a delay equation. On the other hand, modern modeling packages such as MODELICA[1] or MATLAB/SIMULINK[2] automatically generate dynamical systems with constraints that

---

*Institut für Mathematik, Technische Universität Berlin, Str. des 17. Juni 136, 10623 Berlin, Federal Republic of Germany, `{ahrens,unger}@math.tu-berlin.de`.

[1]https://www.modelica.org/

[2]https://www.mathworks.com/

arise due to interface conditions or conservation laws. As a consequence, we have to deal with equations of the form (1.1), which combine the features of *delay differential equations* (DDEs) [2, 3, 11, 16] and *differential-algebraic equations* (DAEs) [20, 24]. Note that DDAEs may also arise from data-driven approaches [9, 30, 31], where the time delay is used to realize a transport phenomenon. Let us mention that our framework is not restricted to a single time delay, since multiple commensurate delays can be rewritten in the form (1.1) by introducing new variables [12].

One of the main difficulties for DDAEs is the fact that solutions may depend on derivatives of the function F in (1.1) as well as on evaluations of F at future time points [6, 13], i.e., one has to study the interplay of the differentiation operator $\mathrm{d}/\mathrm{d}t$ and the (time) shift operator Δ_τ [15], defined via $(\Delta_\tau x)(t) = x(t + \tau)$. In particular, it is important to understand which equations in (1.1) need to be differentiated and which need to be shifted. For linear systems, this is investigated for instance in [6, 13–15, 32, 33]. For nonlinear systems a detailed analysis is not available – we instead refer to [1] for an analysis of (1.1) in a particular Hessenberg form.

In this paper, we propose a graph-theoretical approach to determine which equations need to be differentiated and which equations need to be shifted. To this end, we revisit the *Pantelides* algorithm [23] for DAEs in Section 2, a methodology that uses the information which variable appears in which equation to determine which equations need to be differentiated. Our main contributions are the following:

(a) We introduce equivalence classes (cf. Definition 2.1) in the graph of an equation, which allows us to derive a criterion – see Theorem 2.10 – when the Pantelides algorithm terminates. We show that this criterion is equivalent to the criterion presented in [23] and thus provides another interpretation, that enables us to extend the algorithm to the DDAE case.

(b) In Section 3.1 we illustrate that, from a structural point of view, the operations differentiation and shifting are similar, such that we can essentially use the same procedure to determine the equations that need to be shifted or differentiated, respectively.

(c) We introduce a shifting and differentiation graph in Definition 3.5 and show that differentiation does not affect the shifting graph (Proposition 3.8). As a consequence, we follow the strategy in [6, 32], and first determine the equations that need to be shifted. It turns out that during this procedure some of the equations need to be differentiated and we propose a linear integer program to determine which equations need to be differentiated how many times. Theorem 3.12 details that the integer program is equivalent to a standard linear program that can be solved with standard methods.

(d) Our methodology is summarized in Algorithm 4. We conclude this manuscript with a detailed analysis of Algorithm 4 in Section 4: We show in Theorem 4.1 that Algorithm 4 terminates if and only if the DDAE is structurally nonsingular with respect to a certain equivalence relation. Moreover Theorem 4.4 details in which cases Algorithm 4 concludes that no equations need to be shifted and that in this situation it produces the same result as if we apply the original Pantelides algorithm to the DDAE (1.1) where we replace $\Delta_{-\tau} x$ with a function parameter λ.

We are convinced that our methodology can be combined with a dummy derivative approach [21] or with algebraic regularization techniques as proposed in [28, 29], and thus serves as

the first step to a numerical method for general nonlinear DDAEs. We emphasize that – similar to the original Pantelides algorithm [23] – our method not always determines the correct number of differentiations and shifts. For DAEs this fact is accounted for with a success check [25, 26] and an algebraic analysis [28]. We foresee that a similar validation of the results is also possible for DDAEs.

Notation   The natural numbers, the nonnegative integers, and the reals are denoted by $\mathbb{N}$, $\mathbb{N}_0$, and $\mathbb{R}$, respectively. For a set X the power set of X is denoted by $\mathcal{P}(X)$. For a differentiable function $f \colon \mathbb{I} \to \mathbb{R}^n$ we use the notation $\dot{f} := \frac{\mathrm{d}}{\mathrm{d}t} f$ to denote the derivative from the right[3] with respect to the (time) variable t. Similarly, the shift operator Δ_τ is defined as $(\Delta_\tau f)(t) = f(t + \tau)$. As a consequence, the DDAE (1.1) can be conveniently written as

$$F(t, x, \dot{x}, \Delta_{-\tau} x) = 0.$$

The partial derivative of F with respect to x, $\dot{x}$, and $\Delta_{-\tau} x$ is denoted by $\frac{\partial F}{\partial x}$, $\frac{\partial F}{\partial \dot{x}}$, and $\frac{\partial F}{\partial \Delta_{-\tau} x}$, respectively. The union of two sets A and B is denoted by $A \cup B$. If, moreover, the intersection $A \cap B$ of A and B is empty, then we write $A \dot{\cup} B$. The number of elements that are contained in the set A is denoted by $|A|$. For an equivalence relation $R \subseteq A \times A$ we denote the set of equivalence classes by A/R. For a subset $\widehat{A} \subseteq A$ we define $\widehat{A}/R := \widehat{A}/\widehat{R}$ with

$$\widehat{R} := \{(x, y) \in R \mid x, y \in \widehat{A}\}.$$

2   Preliminary Results

A standard approach to solve differential equations with delay is the method of steps [2]. In the context of DDAEs, this requires to solve a sequence of initial value problems with a DAE, see for instance [33]. More precisely, we study the DAE

$$F(t, x, \dot{x}) = 0, \tag{2.1a}$$

where $F \colon \mathbb{I} \times \mathbb{D}_x \times \mathbb{D}_{\dot{x}} \to \mathbb{R}^m$, $\mathbb{I} \subseteq \mathbb{R}$ is an interval and $\mathbb{D}_x, \mathbb{D}_{\dot{x}} \subseteq \mathbb{R}^n$ are open, together with the initial condition

$$x(t_0) = x_0. \tag{2.1b}$$

One of the main differences of DAEs compared to *ordinary differential equations* (ODEs) is that the partial derivative of F with respect to $\dot{x}$, denoted via $\frac{\partial F}{\partial \dot{x}}$, may be singular. In fact, only if $n = m$ the DAE (2.1a) is well-posed [20], and thus for the remainder of this manuscript we assume $n = m$. It is well-known that in general solutions of (2.1) depend on derivatives of (2.1a), see for instance [4, 20, 24]. Following [5], we introduce the so-called *derivative array* of level ℓ

$$\mathcal{D}_\ell(t, x, \dot{x}, \ldots, x^{(\ell+1)}) := \begin{bmatrix} F(t, x, \dot{x}) \\ \frac{\mathrm{d}}{\mathrm{d}t} F(t, x, \dot{x}) \\ \vdots \\ \frac{\mathrm{d}^\ell}{\mathrm{d}t^\ell} F(t, x, \dot{x}) \end{bmatrix},$$

which can be used to determine the solution of (2.1) for large enough ℓ. The *strangeness index* [19, 20] for instance uses $\mathcal{D}_\ell$ to determine matrix functions, which can be used to

---

[3]For delay equations it is standard to use the derivative from the right [6, 16] since the history function may not be linked smoothly to the solution of the associated initial value problem.

construct a reformulated version of (2.1a) that is suitable for numerical time integration. Although this procedure is tailored to numerical methods, it suffers from two issues: First, in a large-scale setting, the computation of the matrix functions may become computationally expensive, since all equations in (2.1a) are differentiated instead of only the required equations. Second, the computation of the matrix functions requires several numerical challenging rank decisions.

To overcome these issues, [28,29] propose to combine such algebraic regularization techniques with so-called structural analysis tools, such as the Σ-method [25,26] and the Pantelides algorithm [23]. Hereby, structure is understood as the information which variable appears in which equations. In other words, the structure defines a bipartite graph $G = (V_\mathrm{E} \dot\cup V_\mathrm{V}, E)$ where the set of vertices $V_\mathrm{E} \dot\cup V_\mathrm{V}$ is given by the set of equations V_E and the set of variables V_V. An edge $\{F, x\} \in E$ implies that the variable $x \in V_\mathrm{V}$ appears in the equation $F \in V_\mathrm{E}$ (see Definition 2.1). Note that we need to distinguish between a variable and the name of the variable in the definition of the bipartite graph. We, therefore, introduce the language

$$L := \left\{ \Delta_{k\tau} \xi_j^{(i)} \mid \xi \in \{F, x\},\, j \in \mathbb{N},\, i \in \mathbb{N}_0,\, k \in \{-1\} \cup \mathbb{N}_0 \right\},$$

which accounts for differentiation and also for shifting, which is required for the upcoming analysis of the DDAE (1.1). For notational convenience we write ξ_j instead of $\xi_j^{(0)}$, $\dot\xi_j$ instead of $\xi_j^{(1)}$ and $\xi_j^{(i)}$ instead of $\Delta_{0\tau} \xi_j^{(i)}$. To map a vector of variable or equation names (as defined in L) to a set containing these elements, we define

$$\mathrm{set} \colon L^N \to \mathcal{P}(L), \qquad (a_1, \ldots, a_N) \mapsto \{a_1, \ldots, a_N\}.$$

For the remainder of this paper we identify a variable by its name and do not distinguish between both.

In the study of DDAEs it is essential to group several variables into one single vertex (see the forthcoming Section 3). We, therefore, introduce the following notation.

Definition 2.1. *Consider the equation*

$$F(t, \vartheta) = 0 \tag{2.2}$$

with $F \colon \mathbb{I} \times \mathbb{D}_\vartheta \to \mathbb{R}^M$ and $\mathbb{D}_\vartheta \subseteq \mathbb{R}^N$.

(a) The set

$$\Theta := \left\{ \theta \in \mathrm{set}(\vartheta) \,\Big|\, \frac{\partial F}{\partial \theta} \not\equiv 0 \right\}$$

is called the set of variables *of the equation* (2.2).

(b) Let Θ denote the set of variables of (2.2) and $\widetilde\Theta \subseteq \Theta$. The graph *of (2.2) over the equivalence relation $R \subseteq \widetilde\Theta \times \widetilde\Theta$ is defined to be the bipartite graph $G = (V_\mathrm{E} \dot\cup V_\mathrm{V}, E)$ where $V_\mathrm{E} := \mathrm{set}(F)$ is the set of equations, $V_\mathrm{V} := \widetilde\Theta / R$ denotes the set of variables and*

$$E := \left\{ \{V, w\} \,\Big|\, V \in V_\mathrm{V}, w \in V_\mathrm{E} : \text{there exists } v \in V \text{ such that } \frac{\partial w}{\partial v} \not\equiv 0 \right\}. \tag{2.3}$$

(c) Let Θ denote the set of variables for the DAE (2.1a) (with $\vartheta^T = \begin{bmatrix} x^T & \dot x^T \end{bmatrix}$) and $R := \{(\theta, \theta) \mid \theta \in \Theta\} \subseteq \Theta \times \Theta$. The bipartite graph $G = (V_\mathrm{E} \dot\cup V_\mathrm{V}, E)$ is called graph of the DAE (2.1a) *if G is the graph of (2.1a) over R.*

4

(a) Graph of the DAE

(b) Assignment after deleting x_1 and x_2

(c) Graph for (2.4) after differentiating F_1

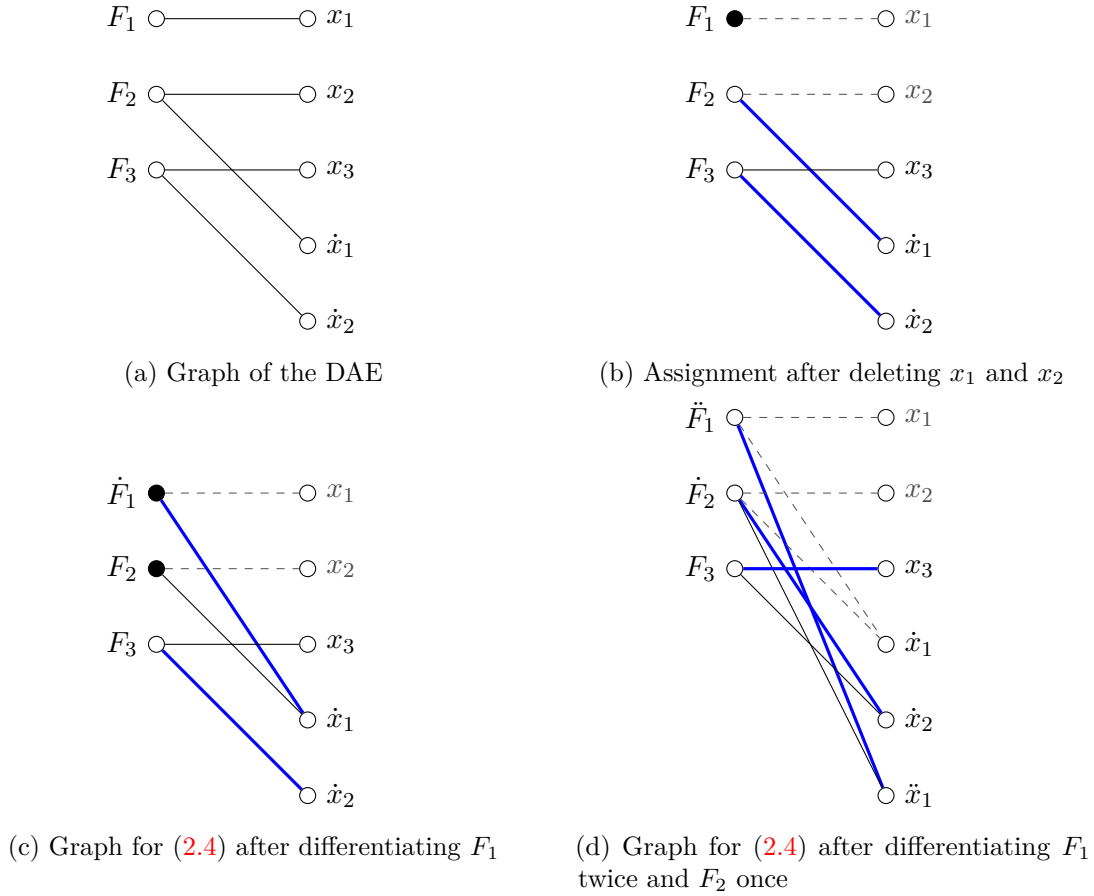(d) Graph for (2.4) after differentiating F_1 twice and F_2 once

Figure 2.1: Visualization of Example 2.2

Example 2.2. In the DAE

$$0 = x_1 + f_1, \qquad \dot{x}_1 = x_2 + f_2, \qquad \dot{x}_2 = x_3 + f_3 \qquad (2.4)$$

we denote the ith equation by F_i for $i = 1, 2, 3$. The graph $G = (V_\mathrm{E} \dot\cup V_\mathrm{V}, E)$ of the DAE (2.4) given by

$$V_\mathrm{E} := \{F_1, F_2, F_3\} \subseteq \mathcal{P}(L),$$
$$V_\mathrm{V} := \{\{x_1\}, \{x_2\}, \{x_3\}, \{\dot{x}_1\}, \{\dot{x}_3\}\} \subseteq \mathcal{P}(L),$$
$$E := \{\{F_1, \{x_1\}\}, \{F_2, \{x_2\}\}, \{F_2, \{\dot{x}_1\}\}, \{F_3, \{x_3\}\}, \{F_3, \{\dot{x}_2\}\}\}$$

is visualized in Figure 2.1a (where we represent the equivalence classes by listing all its elements). By standard abuse of notation we do not distinguish between a graph and its visualization. ◯

The main idea of the Pantelides algorithm [23] is to answer the question which equation determines which variable. More precisely, we are interested in matching each equation with a *highest derivative*, i.e., a variable $x_i^{(k)}$ such that $x_i^{(k)}$ occurs in this equation but $x_i^{(k+\ell)}$ for $\ell > 0$ does not occur in any equation. For instance, in an ODE of the form $\dot{x} = f(t, x)$ we can simply assign the ith equation to $\dot{x}_i$.

In the language of graph theory, we are thus interested in a *matching* (in the literature also called *assignment*) $M \subseteq E$ of a graph (V, E), which is a subset of edges, such that for all

vertices $v \in V$ with $v \in e_1 \in M$ and $v \in e_2 \in M$ we have $e_1 = e_2$, i.e., no vertex appears in more than one edge. An edge $e = \{v_1, v_2\} \in M$ is called a *matching edge* and we say that v_1 and v_2 are *matched*. A vertex which does not occur in any matching edge is called an *exposed vertex* with respect to the matching. In a bipartite graph $G = (V_E \dot{\cup} V_V, E)$ a *maximal matching* is a matching such that no $F \in V_E$ is exposed.

Example 2.3. In the graph in Example 2.2 the variables x_1, x_2 are not the highest derivative such that we can (temporarily) delete these variables from the graph, which we visualize by using gray edges in Figure 2.1b. A matching is for instance given by $M = \{\{F_2, \{\dot{x}_1\}\}, \{F_3, \{\dot{x}_2\}\}\}$, which we depict in Figure 2.1b via thick blue line. We immediately notice that F_1 is an exposed vertex (independent of any possible matching and indicated by a black vertex) and thus no maximal matching is possible. ◯

Remark 2.4. Let Θ denote the set of variables for the DAE (2.1a) (cf. Definition 2.1). Define

$$\widetilde{\Theta} := \{\theta \in \Theta \mid \theta \text{ is highest derivative}\} \tag{2.5}$$

and $\widetilde{R}_{\text{equal}} := \{(\theta, \theta) \mid \theta \in \widetilde{\Theta}\}$. Then the graph of the DAE that is obtained after deleting all variables that are not highest derivative is the graph of (2.1a) over $\widetilde{R}_{\text{equal}}$. ◇

Suppose that, after deleting variables that are no highest derivatives, we cannot find a maximal matching and thus have an exposed vertex F_i. Then one of the following applies: The function F_i does not depend on any variable, implying that in general, the DAE is not regular (see [19] for further details). Otherwise, the function F_i determines a variable that is not a highest derivative, and thus F_i needs to be differentiated to construct a solution. More precisely, we replace F_i by $\frac{\mathrm{d}}{\mathrm{d}t} F_i(t, \vartheta, \dot{\vartheta}) = 0$ and add possible new variables. Note that all variables that appear in F_i appear with one additional derivative in $\frac{\mathrm{d}}{\mathrm{d}t} F_i(t, \vartheta, \dot{\vartheta}) = 0$ and thus the original variables cannot be highest derivatives and can thus be ignored in the following.

Example 2.5. Continuing with Example 2.3, we replace F_1 with the differentiated equation $\frac{\mathrm{d}}{\mathrm{d}t} F_1(t, \vartheta, \dot{\vartheta}) = 0$, which we denote by $\dot{F}_1$. Note that there is a (gray) edge between $\dot{F}_1$ and x_1, since in our structural approach we do not work with the actual equation but only with the information that F_1 depends on x_1. The chain rule then implies that $\dot{F}_1$ depends on x_1 and $\dot{x}_1$. A possible matching is given by $M = \{\{\dot{F}_1, \{\dot{x}_1\}\}, \{F_3, \{\dot{x}_2\}\}\}$, such that F_2 is an exposed vertex with respect to M. The resulting graph is depicted in Figure 2.1c. Continuing with our strategy for Example 2.5 implies that we need to differentiate equation F_2. This time, the situation is slightly different than before, since there exists a coupling between $\dot{F}_1$ and F_2 via the variable $\dot{x}_1$ indicated by a black vertex, i.e., we have a path from $\dot{F}_1$ to F_2 via $\dot{x}_1$. ◯

A *path* is a sequence of edges $(e_1, \dots, e_n)$ in a graph $G = (V, E)$ such that $e_i = \{v_{i-1}, v_i\} \in E$, where $v_0, \dots, v_n \in V$ are distinct. An *alternating path* with respect to a matching is a path with alternating non-matching and matching edges that starts with a non-matching edge. An *augmenting path* with respect to a matching is an alternating path that ends with a non-matching edge. It is easy to see that whenever an augmenting path exists, we can find another matching that includes more equations than the previous matching and hence we cannot determine (at this point) that the equation needs to be differentiated. Hence we only have to differentiate whenever we cannot find an augmenting path. In this case, we

have to differentiate all equations that are connected with the exposed equation via an alternating path. For further details, we refer to [23].

Example 2.6. In Example 2.5 there exists no augmenting path that starts in F_2 (cf. Figure 2.1c). Thus, we have to differentiate F_2 and all equations F_j with an alternating path from F_2 to F_j. In Figure 2.1c these equations are denoted by a black dot. In this case the path $(F_2, \{\dot{x}_1\}, \dot{F}_1)$ implies that we have to differentiate $\dot{F}_1$ and F_2. Note that in the resulting graph, given in Figure 2.1d, we find the maximal matching

$$M = \{\{\ddot{F}_1, \{\ddot{x}_1\}\}, \{\dot{F}_2, \{\dot{x}_2\}\}, \{F_3, \{x_3\}\}\}$$

and can thus stop. ◯

Before we present an algorithmic summary of the outline methodology, let us recall the main idea of the Pantelides algorithm, namely to match each equation in (2.1a) with a highest derivative. In more detail, we have considered the highest derivative of each variable (for the corresponding equivalence relation see Remark 2.4) to determine which equations need to be differentiated. Let $\{k_1, \ldots, k_m\} \subseteq \{1, \ldots, M\}$ denote the equations that need to be differentiated. Then we define a *subset of equations* of (2.2) via

$$\widehat{F}(t, \widehat{\vartheta}) := \begin{bmatrix} F_{k_1}(t, \vartheta) \\ \vdots \\ F_{k_M}(t, \vartheta) \end{bmatrix} = 0, \tag{2.6}$$

with the understanding that $\widehat{\vartheta}$ contains only the subset of variables of ϑ that satisfies

$$\frac{\partial \widehat{F}}{\partial \widehat{\vartheta}} \not\equiv 0.$$

It is clear that we cannot match each equation in (2.6) with a highest derivative if there are more equations than highest derivative variables, i.e., equivalence classes in $\left(\text{set}(\hat{\vartheta}) \cap \tilde{\Theta}\right)/R_{\text{equal}}$ with $\tilde{\Theta}$ defined as in (2.5).

Definition 2.7. *A subset* (2.6) *of* (2.2) *is called* structurally singular *with respect to an equivalence class* $\widehat{\Theta}/\widehat{R}$ *with* $\widehat{\Theta} \subseteq \text{set}(\hat{\vartheta})$ *and relation* $\widehat{R}$ *if*

$$|\text{set}(\widehat{F})| > |\widehat{\Theta}/\widehat{R}|.$$

The equation (2.2) *is called* structurally singular *with respect to an equivalence class* Θ/R *with* $\Theta \subseteq \text{set}(\vartheta)$ *if it contains a structurally singular subset* (2.6) *with respect to* $(\Theta \cap \text{set}(\hat{\vartheta}))/R$[4]. *A structurally singular subset* (2.6) *is called* minimal structural singular *(MSS) with respect to an equivalence class* Θ/R *if none of its proper subsets* (2.6) *is structurally singular with respect to* $(\Theta \cap \text{set}(\hat{\vartheta}))/R$.

Remark 2.8. The notion of structural singularity of a subset of equations is defined in [23]. In contrast to [23] we restrict ourselves to the set of variables that are actually contained within this subset of equations. ◇

---

[4]Recall that we use $(\Theta \cap \text{set}(\hat{\vartheta}))/R$ to denote the set of equivalence classes $(\Theta \cap \text{set}(\hat{\vartheta}))/\widehat{R}$ with the restricted equivalence relation $\widehat{R} := \{(x, y) \in R \mid x, y \in \Theta \cap \text{set}(\hat{\vartheta})\}$

Proposition 2.9. *Consider the equation* (2.2), *let* Θ *denote the set of variables of* (2.2). *For a subset* $\tilde{\Theta} \subseteq \Theta$ *with equivalence relation* $R \subseteq \tilde{\Theta} \times \tilde{\Theta}$ *consider the graph* G *of* (2.2) *over* $\tilde{\Theta}/R$ *with a matching* M. *Let* F_j *be an exposed node. Then one of the following is true:*

(a) *There exists an augmenting path starting in* F_j.

(b) *The set of equations* $\widehat{F}(t, \widehat{\vartheta}) = 0$ *associated with*

$$C_{F_j} := \{F_k \in V_E \mid \text{there exists an alternating path between } F_j \text{ and } F_k \text{ in } G\} \quad (2.7)$$

is MSS with respect to $(\tilde{\Theta} \cap \text{set}(\widehat{\vartheta}))/R$.

Proof. Since all equations in C_{F_j} have an alternating path to F_j we conclude that F_j is the only exposed equation in C_{F_j}. This implies

$$|C_{F_j}| - 1 \leq |(\tilde{\Theta} \cap \text{set}(\widehat{\vartheta}))/R| \quad (2.8)$$

Suppose that there is no augmenting path that starts in F_j. Assume that the inequality in (2.8) is strict. Then there exists at least one equivalence class in $(\tilde{\Theta} \cap \text{set}(\widehat{\vartheta}))/R$ that is exposed. By definition of $(\tilde{\Theta} \cap \text{set}(\widehat{\vartheta}))/R$ and C_{F_j} this implies that there exists an augmenting path that starts in F_j, a contradiction. We conclude that we have equality in (2.8) and thus C_{F_j} is structurally singular.

Consider a proper subset $\tilde{C}_{F_j} \subsetneq C_{F_j}$. If $F_j \notin \tilde{C}_{F_j}$ then all $F_k \in \tilde{C}_{F_j}$ are contained in a matching edge. Thus the set is structurally nonsingular. If $F_j \in \tilde{C}_{F_j}$ there exists an edge between a variable $x_k \in (\tilde{\Theta} \cap \text{set}(\widehat{\vartheta}))/R$ and an equation $F_k \in \tilde{C}_{F_j}$ which is not a matching edge. Thus $\tilde{C}_{F_j}$ is structurally nonsingular. This completes the proof. $\qquad\square$

Proposition 2.9 allows us to implement the outlined strategy as follows. Suppose that we have already found a matching for the first $j-1$ equations. For the jth equation, we try to find an augmenting path. If there exists an augmenting path, then we can directly generate a matching that includes the first j equations. If not, then we differentiate all equations within the set C_{F_j}, see (2.7). Note that, to determine that there is no augmenting path, we have to check all alternating paths and hence the set C_{F_j} is automatically generated when we check for an augmenting path. One way to achieve this is via a recursive, depth-first search algorithm [7, 34]. The details are given in Algorithm 1. Here, we use the notation $\mathtt{x_i}, \mathtt{F_j}$ on the one hand as vertices in V_V and V_E and on the other hand as indices referring to those vertices. The complete algorithm, which is known as the Pantelides algorithm [23], is presented in Algorithm 2. Note that in contrast to the original algorithm [23] we replace equations that are differentiated and do not add the equations to the graph. This has the advantage that we do not need to keep track which equation was obtained by differentiating another equation, and similarly for the variables. In Algorithm 2 we use a similar notation as in Algorithm 1 for vertices in V_V and V_E. The vertex $\mathtt{x_{i,k}}$ represents the variable $x_i^{(k)}$.

Since Algorithm 2 is an iterative algorithm we have to answer the questions whether Algorithm 2 terminates. We immediately observe that if Algorithm 2 terminates, then we have no exposed equation and thus each equation is matched with a highest derivative. We conclude that a necessary condition for termination of Algorithm 2 is that the DAE (2.1a)

Algorithm 1 Augmentpath

Input: bipartite graph $\mathtt{G} = (\mathtt{V_E} \dot{\cup} \mathtt{V_V}, \mathtt{E})$ with $\mathtt{V_E} = \{\mathtt{F_1}, \dots, \mathtt{F_m}\}$ and $\mathtt{V_V} = \{\mathtt{x_1}, \dots, \mathtt{x_n}\}$
    $\mathtt{F_j} \in \mathtt{V_E}$ ▷ starting vertex
    `assign`   ▷ matching
    `colorV, colorE`   ▷ vertices needed to be differentiated if `pathfound== false`
Output: `pathfound, assign, colorV, colorE, path`

 1: `pathfound` ← `false`
 2: `colorE(F`$_\mathtt{j}$`)` ← `true`
 3: **for** $\mathtt{x_i} \in \mathtt{V_V}$ with $\{\mathtt{x_i}, \mathtt{F_j}\} \in \mathtt{E}$ and `colorV(`$\mathtt{x_i}$`) ==` `false` **do**
 4:    `colorV(`$\mathtt{x_i}$`)` ← `true`
 5:    $\mathtt{F_k}$ ← `assign(`$\mathtt{x_i}$`)`
 6:    **if** $\mathtt{F_k} == 0$ **then**
 7:        `pathfound` ← `true`
 8:    **else**
 9:        `(pathfound, assign, colorV, colorE)` ← `Augmentpath(G,` $\mathtt{F_k}$`, assign, colorV,`
    `colorE)`
10:    **end if**
11:    **if** `pathfound == true` **then**
12:        `assign(`$\mathtt{x_i}$`)` ← $\mathtt{F_j}$
13:        **return** `pathfound, assign, colorV, colorE`
14:    **end if**
15: **end for**

---

is structurally nonsingular with respect to $\Theta/R_{\mathrm{equal}}$ with Θ denoting the set of variables of (2.1a) and R_{equal} the equivalence relation

$$R_{\mathrm{equal}} := \left\{ (\theta, \tilde{\theta}) \in \Theta \times \Theta \ \middle|\ \exists \xi \in \mathrm{set}(x),\ p, q \in \mathbb{N}_0 \text{ such that } \theta = \xi^{(p)},\ \tilde{\theta} = \xi^{(q)} \right\}. \quad (2.9)$$

The following result shows that in fact the structural nonsingularity is also a sufficient condition.

Theorem 2.10. *Consider the DAE* (2.1a) *and let* Θ *denote its set of variables and* R_{equal} *the equivalence relation in* (2.9). *Then Algorithm 2 applied to the DAE* (2.1a) *terminates if and only if* (2.1a) *is structurally nonsingular with respect to* $\Theta/R_{\mathrm{equal}}$.

Proof. We show that the structural nonsingularity is equivalent to the termination criterion that is presented in the original paper [23]. To this end let us rewrite (2.1a) by splitting the variable x into $\mathrm{set}(x) = \mathrm{set}(y) \dot{\cup} \mathrm{set}(z)$ such that

- for all $\xi \in \mathrm{set}(y)$ it holds that $\frac{\partial F}{\partial \xi} \not\equiv 0$ and

- for all $\xi \in \mathrm{set}(z)$ it holds that $\frac{\partial F}{\partial \xi} \equiv 0$.

Using the new variables we can rewrite (2.1a) as

$$F(t, y, \dot{y}, z) = 0. \quad (2.10a)$$

We introduce a coupling between y and $\dot{y}$ by adding formal equations

$$G_i(y_i, \dot{y}_i) = 0 \qquad \text{for } i = 1, \dots, n_y \quad (2.10b)$$

9

Algorithm 2 Pantelides Algorithm for DAEs

Input: graph $G = (V_E \dot{\cup} V_V, E)$, $V_E = \left\{ F_{j,\ell} \mid F_j^{(\ell)} \in V_E \right\}$, $V_V = \left\{ x_{i,k} \mid x_i^{(k)} \in V_V \right\}$
Output: graph $G = (V_E \dot{\cup} V_V, E)$

1: $\texttt{assign}(x_{i,k}) \leftarrow 0$ for each $x_{i,k} \in V_V$
2: **for** $r = 1, \ldots, M$ **do**
3: $\quad (p, q) \leftarrow (r, 0)$
4: $\quad$ **repeat**
5: $\qquad \tilde{E} \leftarrow E \setminus \left\{ \{x_{i,k}, F_{j,\ell}\} \in E \mid x_{i,k+s} \in V_V \text{ for some } s > 0 \right\}$
6: $\qquad \tilde{V}_V \leftarrow V_V \setminus \left\{ x_{i,k} \mid x_{i,k+s} \in V_V \text{ for some } s > 0 \right\}$
7: $\qquad \texttt{colorV}(x_{i,k}) \leftarrow \texttt{false}$ for each $x_{i,k} \in \tilde{V}_V$
8: $\qquad \texttt{colorE}(F_{j,\ell}) \leftarrow \texttt{false}$ for each $F_{j,\ell} \in V_E$
9: $\qquad (\texttt{pathfound}, \texttt{assign}, \texttt{colorV}, \texttt{colorE}) \leftarrow \text{Augmentpath}((V_E \dot{\cup} \tilde{V}_V, \tilde{E}), F_{p,q},$
$\qquad\qquad \texttt{assign}, \texttt{colorV}, \texttt{colorE})$
10: $\qquad$ **if** $\texttt{pathfound} == \texttt{false}$ **then**
11: $\qquad\qquad V_V \leftarrow V_V \cup \{x_{i,k+1} \mid \texttt{colorV}(x_{i,k}) == \texttt{true}\}$
12: $\qquad\qquad$ **for each** (j, ℓ) s.t. $\texttt{colorE}(F_{j,\ell}) == \texttt{true}$ **do**
13: $\qquad\qquad\qquad F_{j,\ell} \leftarrow F_{j,\ell+1}$
14: $\qquad\qquad\qquad E \leftarrow E \cup \left\{ \{x_{i,k+1}, F_{j,\ell}\} \mid \{x_{i,k}, F_{j,\ell}\} \in E \right\}$
15: $\qquad\qquad$ **end for**
16: $\qquad\qquad$ **for each** i with $\texttt{colorV}(x_{i,k}) == \texttt{true}$ and $F_{j,\ell} == \texttt{assign}(x_{i,k})$ **do**
17: $\qquad\qquad\qquad \texttt{assign}(x_{i,k+1}) \leftarrow F_{j,\ell}$
18: $\qquad\qquad$ **end for**
19: $\qquad$ **end if**
20: $\quad$ **until** $\texttt{pathfound} = \texttt{true}$
21: **end for**

---

for $y = [y_i]_{i=1,\ldots,n_y}$ and $G = [G_i]_{i=1,\ldots,n_y}$. The system of equations (2.10) is called the *extended DAE*. In [23] it is shown that Algorithm 2 applied to the DAE (2.1a) terminates if and only if the extended DAE (2.10) is structurally nonsingular with respect to $\text{set}(y, \dot{y}, z)/R_{\text{triv}}$ with $R_{\text{triv}} := \{(\xi, \xi) \mid \xi \in \text{set}(y, \dot{y}, z)\}$. It therefore suffices to show that structural nonsingularity of (2.10a) with respect to $\text{set}(y, \dot{y}, z)/R_{\text{equal}}$ is equivalent to structural nonsingularity of the extended DAE (2.10) with respect to $\text{set}(y, \dot{y}, z)/R_{\text{triv}}$.

Assume first that the extended DAE (2.10) is structurally nonsingular with respect to $\text{set}(y, \dot{y}, z)/R_{\text{triv}}$. Consider an arbitrary subset of (2.10a)

$$\widehat{F}(t, \eta, \gamma, \dot{\gamma}, \dot{\mu}, \widehat{z}) = 0, \tag{2.11}$$

with $\text{set}(\eta, \gamma, \mu) \subseteq \text{set}(y)$ and $\text{set}(\widehat{z}) \subseteq \text{set}(z)$. Let $\widehat{G}(\gamma, \dot{\gamma}) = 0$ denote the subset of (2.10b) that corresponds to the variable γ. Since the extended DAE is structurally nonsingular, we obtain

$$\left| \text{set}\left(\widehat{F}, \widehat{G}\right) \right| \leq \left| \text{set}\left(\eta, \gamma, \dot{\gamma}, \dot{\mu}, \widehat{z}\right) / R_{\text{triv}} \right|.$$

This is equivalent to

$$\left| \text{set}\left(\widehat{F}\right) \right| \leq \left| \text{set}\left(\eta, \gamma, \dot{\mu}, \widehat{z}\right) \right| = \left| \text{set}(\eta, \gamma, \dot{\gamma}, \dot{\mu}, \widehat{z})/R_{\text{equal}} \right|$$

and thus (2.10a) is structurally nonsingular with respect to Θ/R_{equal}.

Conversely suppose that the DAE (2.10a) is structurally nonsingular with respect to $\mathrm{set}(y, \dot{y}, z)/R_{\mathrm{equal}}$. A subset of (2.10) is given by

$$\widehat{F}(t, \eta, \widehat{\eta}, \gamma, \dot{\gamma}, \widehat{\gamma}, \dot{\widehat{\gamma}}, \dot{\mu}, \dot{\widehat{\mu}}, \widehat{z}) = 0,$$
$$\widehat{G}(\widehat{\eta}, \dot{\widehat{\eta}}, \widehat{\gamma}, \dot{\widehat{\gamma}}, \widehat{\mu}, \dot{\widehat{\mu}}, \widehat{\zeta}, \dot{\widehat{\zeta}}) = 0. \tag{2.12}$$

with $\mathrm{set}(\eta, \widehat{\eta}, \gamma, \widehat{\gamma}, \mu, \widehat{\mu}, \widehat{\zeta}) \subseteq \mathrm{set}(y)$ and $\mathrm{set}(\widehat{z}) \subseteq \mathrm{set}(z)$. Since the DAE (2.10a) is structurally nonsingular with respect to $\mathrm{set}(y, \dot{y}, z)/R_{\mathrm{equal}}$ we obtain

$$\left| \mathrm{set}\left(\widehat{F}\right) \right| \leq \left| \mathrm{set}\left(\eta, \widehat{\eta}, \gamma, \dot{\gamma}, \widehat{\gamma}, \dot{\widehat{\gamma}}, \dot{\mu}, \dot{\widehat{\mu}}, \widehat{z}\right)/R_{\mathrm{equal}} \right| = \left| \mathrm{set}\left(\eta, \widehat{\eta}, \gamma, \widehat{\gamma}, \dot{\mu}, \dot{\widehat{\mu}}, \widehat{z}\right) \right|$$

which immediately implies

$$\left| \mathrm{set}\left(\widehat{F}, \widehat{G}\right) \right| \leq \left| \mathrm{set}\left(\eta, \widehat{\eta}, \dot{\widehat{\eta}}, \gamma, \widehat{\gamma}, \dot{\widehat{\gamma}}, \dot{\mu}, \widehat{\mu}, \dot{\widehat{\mu}}, \widehat{\zeta}, \dot{\widehat{\zeta}}, \widehat{z}\right) \right|.$$

We conclude that (2.12) is structrually nonsingular with respect to R_{triv}, which completes the proof. □

Even if Algorithm 2 terminates this is not a guarantee that the correct number of differentiations for each equation is determined. In fact, the Pantelides algorithm relies heavily on the so-called sparsity pattern of the DAE, i.e., each equation depends only on few variables. In particular, Algorithm 2 is not invariant under bijective transformations of the DAE (cf. [28, Remark 5]).

Example 2.11. The Pantelides algorithm applied to the DAE

$$\dot{x}_1 + \dot{x}_2 = x_2 + f_1, \qquad 0 = x_1 + x_2 + f_2$$

determines that the second equation needs to be differentiated one time. If we however add the first equation to the second equation, then we can match the first equation with $\dot{x}_1$ and the second equation with $\dot{x}_2$ and thus neither of the equations is differentiated. ○

Also, it may be the case that the Pantelides algorithm concludes that equations need to be differentiated several times, although this is not necessary [27]. One possible reason for this is that the DAE is close to a high-index DAE and thus the numerical solution may be more difficult to obtain than the (potentially) small index suggests (see [27] for further details). In any case, it is possible to check that the equations are differentiated sufficiently often by applying a success check [25, 26] and an algebraic criterion [28].

3 The Pantelides Algorithm for DDAEs

It is well-known that for DDAEs of the form (1.1) it is not sufficient to differentiate equations to obtain solutions, but in addition, some equations need to be shifted [6, 13, 14, 32].

Example 3.1. The first equation in the DDAE

$$\dot{x}_1 = f_1, \qquad 0 = x_1 + \Delta_{-\tau} x_2 + f_2 \tag{3.1}$$

determines $\dot{x}_1$. If we insert the first equation into the differentiated second equation and shift the result we obtain $0 = \Delta_\tau f_1 + \dot{x}_2 + \Delta_\tau \dot{f}_2$ which determines $\dot{x}_2$. ○

As a direct consequence, a structural analysis for DDAEs should reveal which equations need to be shifted and which equations need to be differentiated. We thus have to understand the similarities and differences between differentiating and shifting from a structural point of view.

3.1 Differentiation vs. Shifting

To understand the effect of the two operators $\frac{\mathrm{d}}{\mathrm{d}t}$ and $\Delta_{-\tau}$ from a structural point of view, we start with the following simple example.

Example 3.2. Let $N \in \mathbb{R}^{n \times n}$ be a nilpotent matrix with $N^\nu = 0$ and $N^{\nu-1} \neq 0$ for some $\nu \in \mathbb{N}$ and consider the two equations

$$N\frac{\mathrm{d}}{\mathrm{d}t}x = x + f, \tag{3.2a}$$

$$Ny = \Delta_{-\tau}y + g, \tag{3.2b}$$

which can be understood as special cases of the Weierstraß canonical form [10] for a regular matrix pencil. It is easy to see that the solutions are of the form

$$x = -\sum_{i=0}^{\nu-1} N^i \left(\frac{\mathrm{d}}{\mathrm{d}t}\right)^i f \qquad \text{and} \qquad y = -\sum_{i=0}^{\nu-1} N^i \left(\Delta_\tau\right)^i \Delta_\tau g. \tag{3.2c}$$

Apart from the operator applied to the inhomogeneity both solutions are the same. Thus the variable $\dot{x}$ has in the DAE the same role as the variable y in the difference equation (3.2b). In other words, in the DAE, the variable with the highest derivative is $\dot{x}$, while in the delay equation, the variable $y = \Delta_0 y$ is the variable with the highest shift, in the sense that $0 > -1$. $\qquad\bigcirc$

Motivated by Example 3.2 we can use the Pantelides algorithm (Algorithm 2) for difference equations by replacing differentiation by shifting in Algorithm 2. In a difference equation we want to assign each equation to a different variable with *highest shift*, meaning that $\Delta_{k\tau}x$ with $k \geq 0$ occurs in some equation but $\Delta_{(k+\ell)\tau}x$ for $\ell > 0$ does not. Note that by our definition the variable $\Delta_{-\tau}x$ can never be a variable with *highest shift*, since this would imply that we can solve directly for this variable without shifting. This can also be seen in the solution formula (3.2c) for the difference equation, which requires an additional shift of the inhomogeneity g.

Example 3.3. The graph of the scalar difference equation

$$0 = \Delta_{-\tau}x_1 + f_1 \tag{3.3}$$

is visualized in Figure 3.1a. By replacing differentiating with shifting in Algorithm 2, we

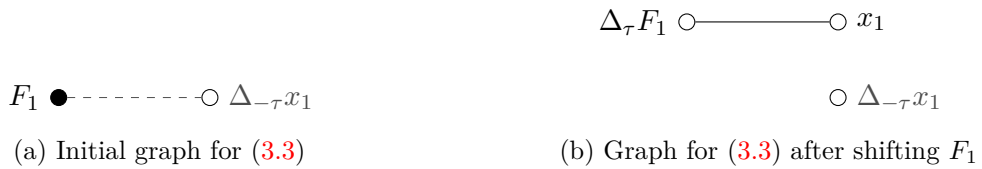(a) Initial graph for (3.3) (b) Graph for (3.3) after shifting F_1

Figure 3.1: Graphs obtained by applying Algorithm 2 to Example 3.3

shift F_1 once (cf. Figure 3.1b). Note that there is no edge between $\Delta_\tau F_1$ and $\Delta_{-\tau}x_1$, since there is no chain rule for the shift operator. $\qquad\bigcirc$

3.2 The Shifting and Differentiation Graph

As outlined in Example 3.1, some equations in the DDAE (1.1) need to be shifted and some need to be differentiated. Motivated by the observation (cf. Section 3.1) that shifting and differentiating are quite similar from a structural point of view, we construct two graphs: One to determine which equations need to be shifted and one to determine which equations need to be differentiated. In the shifting graph, we do not want to match equations with delayed variables, since we cannot (directly) solve for these variables. More precisely, we do not want to match equations with variables that are not highest shifts and thus delete these variables in the shifting graph. On the other hand, we need to make sure that we do not match one equation with a variable and another equation with the derivative of that variable. This is illustrated in the following example.

Example 3.4. Consider again the DDAE (3.1). Deleting the variable $\Delta_\tau x_2$ results in the graph Figure 3.2a with maximal matching $M = \{\{F_1, \{x_1\}\}, \{F_2, \{\dot{x}_1\}\}\}$. Note that the maximal matching is possible, since we match both equations with x_1 respectively its derivative $\dot{x}_1$. If we instead merge the variables x_1 and $\dot{x}_1$ into a single vertex we obtain

(a) without grouping

(b) with grouping

Figure 3.2: Different shifting graphs for Example 3.4

the graph in Figure 3.2b, which indicates that we have to shift some of the equations (in agreement with the observation in Example 3.1. ◯

To ensure that we do not match different equations with the same variable (with different differentiation levels), we need to group the corresponding variables as in Example 3.4. To formalize this, consider the equation

$$F(t, \Delta_{-\tau}x, \Delta_{-\tau}\dot{x}, \ldots, \Delta_{-\tau}x^{(\rho)}, \ldots, \Delta_{\kappa\tau}x, \Delta_{\kappa\tau}\dot{x}, \ldots, \Delta_{\kappa\tau}x^{(\rho)}) = 0. \quad (3.4)$$

which may be understood as a generalization of (1.1) that appears after shifting and/or differentiation of some of the equations.

Definition 3.5. *Consider the DDAE* (3.4) *and let Θ denote the set of variables for* (3.4) *(with $\vartheta^T = \begin{bmatrix} \Delta_{-\tau}x^T & \ldots & \Delta_{\kappa\tau}x^{(\rho)T} \end{bmatrix}$).*

- *The variables $\theta, \widetilde{\theta} \in \Theta$ are called* shifting similar *if there exists $\xi \in \text{set}(x)$ and integers $k \in \mathbb{Z}$, $p, q \in \mathbb{N}_0$ such that $\theta = \Delta_{k\tau}\xi^{(p)}$ and $\widetilde{\theta} = \Delta_{k\tau}\xi^{(q)}$. Shifting similarity introduces an equivalence relation on Θ, which we denote by R_{shift}. The graph of* (3.4) *over R_{shift} is called* shifting graph.

- *The variables $\theta, \widetilde{\theta} \in \Theta$ are called* differential similar *if there exists $\xi \in \text{set}(x)$ and integers $p, k, \ell \in \mathbb{N}_0$ such that $\theta = \Delta_{k\tau}\xi^{(p)}$ and $\widetilde{\theta} = \Delta_{\ell\tau}\xi^{(p)}$. Differential similarity introduces an equivalence relation on $\widetilde{\Theta}$ with*

$$\widetilde{\Theta} = \{\theta \in \Theta \mid \text{there exists } x \in \Theta, k, p \in \mathbb{N}_0 \text{ such that } \theta = \Delta_{k\tau}x^{(p)}\},$$

13

which we denote by R_{diff}. The graph of (3.4) *over R_{diff} is called* differentiation graph.

Example 3.6. The shifting graph for the DDAE (3.1) is given in Figure 3.2b. For the differentiation graph we have

$$\widetilde{\Theta} = \{\{x_1\}, \{\dot{x}_1\}\},$$

such that the differentiation graph is given in Figure 3.3. Note that $\{x_2\} \notin \widetilde{\Theta}$. ◯
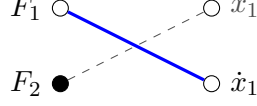

Figure 3.3: Differentiation graph for the DDAE (3.1)

A question that arises is how differentiation of equations affects the shifting graph and how shifting affects the differentiation graph (cf. [15] for an example where the order of shifting and differentiation results in different smoothness requirements). We immediately observe that the differentiation graph is affected by shifting, see the following example.

Example 3.7. The differentiation graph for the scalar DDAE

$$0 = \Delta_{-\tau} x_1 + f_1 \tag{3.5}$$

does not contain any variable vertices and thus in particular no maximal matching. If we shift (3.5) we obtain

$$0 = x_1 + \Delta_\tau f_1,$$

such that the differentiation graph is given by $V = \{F_1, \{x_1\}\}$ and $E = \{\{F_1, \{x_1\}\}\}$. ◯

To understand the effect of differentiation on the shifting graph, let us define the system of equations

$$\check{F}(t, \Delta_{-\tau} x, \dots, \Delta_{-\tau} x^{(\rho)}, \Delta_{-\tau} x^{(\rho+1)}, \dots, \Delta_{\kappa\tau} x, \Delta_{\kappa\tau} \dot{x}, \dots, \Delta_{\kappa\tau} x^{(\rho+1)}) = 0. \tag{3.6}$$

by replacing the ith equation ($i \in \{1, \dots, M\}$) in (3.4) with its total derivative with respect to t. The shifting graphs of (3.4) and (3.6) are denoted by $G^{\text{s}} := (V_{\text{E}}^{\text{s}} \dot\cup V_{\text{V}}^{\text{s}}, E^{\text{s}})$ and $\check{G}^{\text{s}} := (\check{V}_{\text{E}}^{\text{s}} \dot\cup \check{V}_{\text{V}}^{\text{s}}, \check{E}^{\text{s}})$, respectively. Let us define the bijection $\phi \colon V_{\text{E}}^{\text{s}} \dot\cup V_{\text{V}}^{\text{s}} \to \check{V}_{\text{E}}^{\text{s}} \dot\cup \check{V}_{\text{V}}^{\text{s}}$ via

$$\phi(v) = \begin{cases} v \cup \{\dot{w} \mid w \in v\}, & \text{for } v \in V_{\text{V}}^{\text{s}}, \\ v, & \text{for } v \in V_{\text{E}}^{\text{s}} \text{ and } v \neq F_i, \\ \dot{v}, & \text{for } v \in V_{\text{E}}^{\text{s}} \text{ and } v = F_i. \end{cases}$$

Then it is easy to see, that $\{u, v\} \in E^{\text{s}}$ if and only if $\{\phi(u), \phi(v)\} \in \check{E}^{\text{s}}$, that is the graphs G^{s} and $\check{G}^{\text{s}}$ are isomorphic [17, p. 21]. In particular, we have shown the following result.

Proposition 3.8. *Consider the DDAE* (3.4). *Differentiation of an equation in* (3.4) *does not affect the shifting graph of* (3.4).

Since shifting affects the differentiation graph but differentiation does not affect the shifting graph, we conclude that we first determine which equations need to be shifted and afterwards determine which equations should be differentiated.

Remark 3.9. The strategy to shift first can be understood as replacing the DDAE (1.1) with a transformed DDAE that can be solved with the method of steps. For linear time-invariant DDAEs such a strategy was investigated in [6, 32] in a polynomial matrix framework. There the authors use a sequence of row compressions to determine which equations need to be shifted. It should be noted that the row-compression requires differentiation of some of the equations to enable the shift afterward. Fore more details we refer to [6, 32] and the upcoming subsection. ◇

3.3 Differentiating during the Shifting Step

If we find (for instance via Algorithm 2) an exposed equation in the shifting graph, we conclude that this equation and all equations that are connected with this equation by an alternating path need to be shifted. However, simply shifting all these equations may not be sufficient, since the connection may exist only implicitly via the equivalence class. We illustrate this with the following example.

Example 3.10. The shifting graph of Example 3.4, i.e., the shifting graph for the DDAE (3.1), is given in Figure 3.2b. If we match the first equation F_1 with the equivalence class $\{x_1, \dot{x}_1\}$ then F_2 is an exposed equation that is connected with F_1 via $\{x_1, \dot{x}_1\}$. Accordingly, we have to shift both equations. If we check the graph of (3.1) (presented in Figure 3.2a), then we observe that there is no path from F_1 to F_2, i.e., we only obtain a path if we differentiate F_2. ○

To resolve an (possibly) implicit connection, we have to ensure that there also exists a path in the graph of the DDAE and not only in the shifting graph. This can be achieved by differentiating the equation that does not depend on the highest derivative in the equivalence class. Let F_{i_k} for $k = 2, \ldots, K$ denote the equations which have an alternating path to the exposed equation F_{i_1} and let ν_k denote the number how often we have to differentiate the i_kth equation. For each path[5] $(F_{i_{k_j}}, v_j, F_{i_{k_{j+1}}})$ in the shifting graph $(j = 1, \ldots, J)$ with $v_j \in V_V^s$ define $b_j \in \mathbb{Z}$ to be the number how many more times we have to differentiate $F_{i_{k_j}}$ than $F_{i_{k_{j+1}}}$ such that there exists a direct connection to a highest derivative of both equations in the graph of the DDAE. This results in the linear system

$$\nu_{k_j} - \nu_{k_{j+1}} = b_j \qquad j = 1, \ldots, J. \tag{3.7}$$

which we compactly write as $A\nu = b$ with $\nu = [\nu_k] \in \mathbb{N}_0^K$, $b = [b_j] \in \mathbb{Z}^J$ and $A \in \mathbb{Z}^{J \times K}$. Since we do not want to differentiate equations more than necessary, we have to solve the linear integer program

$$\min \sum_{k=1}^{K} \nu_k, \tag{3.8}$$
$$\text{such that } A\nu = b, \quad \nu \in \mathbb{N}_0.$$

Example 3.11. Consider the DDAE

$$\dot{x}_1 = f_1, \qquad 0 = x_2 + f_2, \qquad 0 = x_1 + x_2 + \Delta_{-\tau}x_3 + f_3 \tag{3.9}$$

with shifting graph given in Figure 3.4. Since there exists no maximal matching, we choose F_3 as exposed vertex and obtain that all equations need to be shifted. Since F_1 depends on

---

[5]We do not consider the inverse path, that is the path $(F_{i_{k_{j+1}}}, v_j, F_{i_{k_j}})$, since this results in a redundant equation in the forthcoming linear system.
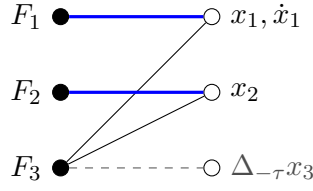
Figure 3.4: The shifting graph for the DDAE (3.9)

$\dot{x}_1$ and F_3 on x_1, we have to differentiate F_3 one more time than F_1. Similarly, we obtain that we have to differentiate F_3 and F_2 the same number of times. Accordingly, the linear integer program (3.8) is given via

$$A = \begin{bmatrix} -1 & 0 & 1 \\ 0 & -1 & 1 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

with the understanding that ν_k determines how often we have to differentiate the F_kth equation. It is easy to see that in this case $\nu_1 = 0$, $\nu_2 = 1$, and $\nu_3 = 1$ is the unique solution of (3.8), i.e. we have to differentiate the second and the third equation. ◯

A relaxation of the linear integer program (3.8) is the linear program

$$\min \sum_{k=1}^{K} \nu_k, \tag{3.10}$$
$$\text{such that } A\nu = b, \quad \nu \geq 0.$$

In fact, we have the following result.

Theorem 3.12.

(a) *The linear program* (3.10) *has a solution if and only if the set*

$$\{\nu \in \mathbb{R}^K \mid A\nu = b\} \tag{3.11}$$

is not empty. In this case the solution is unique.

(b) *The linear integer program* (3.8) *is solvable if and only if the linear program* (3.10) *is solvable. In this case, the minimizing vector $\nu^\star$ of* (3.10) *is the unique minimizer of* (3.8).

Proof.

(a) Recall that J is the number of equations in the linear system $A\nu = b$ and K is the number of unknowns. We immediately observe $J \geq K - 1$ (the equations F_{i_k} form a connected graph) and $\text{rank}(A) \geq K - 1$. Let $e := \begin{bmatrix} 1 & \ldots & 1 \end{bmatrix}^T \in \mathbb{R}^K$. Then

$$Ae = 0$$

and thus we conclude that $\text{rank}(A) = K - 1$. Suppose that the set (3.11) is not empty, i.e., there exists $\widehat{\nu} = [\widehat{\nu}_k] \in \mathbb{R}^K$ such that $A\widehat{\nu} = b$. Define $\alpha^\star = -\min\{\widehat{\nu}_k \mid k = 1, \ldots, K\}$. Then the feasible set of (3.10) is

$$\{\widehat{\nu} + \alpha e \mid \alpha \geq \alpha^\star\}. \tag{3.12}$$

We conclude that $\nu^\star = \widehat{\nu} + \alpha^\star e$ is the unique minimizer of (3.10).

16

(b) Using (a) it suffices to show that (3.8) is solvable whenever (3.10) is solvable. Let $\nu^\star = [\nu_k^\star]$ denote the unique minimizer of (3.10). Then the proof of (a) implies $\min\{\nu_k^\star \mid k = 1, \ldots, K\} = 0$. Since the equations (3.7) imply that the difference between two entries of $\nu^\star$ is an integer, we conclude $\nu^\star \in \mathbb{N}_0^K$. The result follows from the observation that the feasible set of (3.8) is given by

$$\{\nu^\star + \alpha e \mid \alpha \in \mathbb{N}_0\}. \qquad \square$$

Theorem 3.12 implies that we can compute the solution (provided it exists) of the linear integer program (3.8) with standard methods such as the simplex algorithm or interior-point methods [18, 22]. Alternatively, we can exploit the structure of the feasible set (3.12) and simply compute one solution of the linear system $A\nu = b$ and shift the solution accordingly.

Unfortunately, the following examples show that we cannot guarantee that the feasible set is non-empty and hence that the linear integer program (3.8) is solvable.

Example 3.13. Consider the DDAE

$$\dot{x}_1 = f_1, \qquad \dot{x}_1 = x_2 + f_2, \qquad 0 = x_1 + x_2 + \Delta_{-\tau} x_3 + f_3, \qquad (3.13)$$

which can be obtained from (3.9) by adding the first equation to the second equation and renaming of f_2. However, the additional variable in the second equation implies that we have to solve (3.8) with

$$A = \begin{bmatrix} 0 & -1 & 1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix} \qquad \text{and} \qquad b = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}.$$

The first and the third equation imply that $\nu_1 = \nu_2 = \nu_3$, which poses a contradiction to the second equation and thus (3.8) is not solvable. $\qquad \bigcirc$

The reason that the linear integer program (3.8) is not always solvable is due to the fact that in order to resolve the implicit connection via the equivalence we need (at least) one equation with different levels of differentiation (see Example 3.13). Since our main goal is to determine how often we have to shift and differentiate each equation, we are interested only in the maximal number of differentiations that is required. This number may be obtained by determining the equations that are linearly dependent and then solve (3.8) by removing some of the linear dependent equations until (3.8) is solvable. Repeating this procedure with every possible combination of linear dependent equations yields a family of solutions and we can simply take the maximal value for each entry of ν.

Example 3.14. For the linear system in Example 3.13 we observe that the three equations are linearly dependent and that it is sufficient to remove one of them. If we remove the first equation, we obtain $\nu = (0, 0, 1)$, if we remove the second equation, we obtain $\nu = (0, 0, 0)$ and if we remove the third equation we obtain $\nu = (0, 1, 1)$. As a consequence, we have to differentiate the second and the third equation (in agreement with Example 3.11). $\qquad \bigcirc$

We conjecture that if the linear system $A\nu = b$ cannot be solved, then one can determine a-priori, which equations need to be removed to obtain the correct solution. This would allow reducing the computational cost significantly.

3.4 The Pantelides Algorithm for DDAEs

Before we summarize our findings in form of an algorithm, let us briefly recap the results of this section.

- We illustrated that we can use the original Pantelides algorithm [23] (see Algorithm 2) with a different equivalence class to determine which equation needs to be shifted. Recall that we may have to differentiate some equations during the shifting step to resolve implicit connections via equivalence classes (cf. Section 3.3). To account for the additional differentiation, we present a slight modification of Algorithm 2 in Algorithm 3. We emphasize that the notation $\xi_{i,k}$ for $\xi \in \{F, x\}$ has a different meaning for shifting and differentiation. In the case of shifting, the k represents the order of shift while in the case of differentiation k represents the order of differentiation. This enables us to shift in the shifting graph and differentiate in the differentiation graph with the same notation.

- Due to the structure of the equivalence classes, we shift first and then differentiate, see Proposition 3.8. Note that we do not have to resolve implicit connections within the differentiation step, since we only determine the maximal number that each equation needs to be shifted and differentiated. Since a possible implicit connection in the differentiation step results from equations that are already shifted, the implicit connection is resolved automatically by using the same equation with a smaller shift than the maximal shift that was determined in the shifting step.

The complete methodology is summarized in Algorithm 4.

Remark 3.15. During the shifting and differentiation step, we have to update the graph and the corresponding shifting and differentiation graphs. To unify this procedure, the necessary updates of the graph are summarized in Algorithm 5. This algorithm is called in the modified Pantelides algorithm (Algorithm 3). ◇

Example 3.16. The initial shifting graph of the DDAE

$$
\begin{aligned}
0 &= x_1 + f_1 \\
\dot{x}_1 &= \Delta_{-\tau} x_2 + f_2 \\
\dot{x}_2 &= x_1 + \Delta_{-\tau} x_3 + f_3 \\
\dot{x}_3 &= x_4 + \Delta_{-\tau} x_1 + f_4
\end{aligned}
\tag{3.14}
$$

is given in Figure 3.5a and in the notation of Algorithm 4 by the sets

$$
\begin{aligned}
V_{\mathrm{E}}^{\mathrm{s}} &:= \{F_{1,0}, F_{2,0}, F_{3,0}, F_{4,0}\}, \\
V_{\mathrm{V}}^{\mathrm{s}} &:= \{x_{1,0}, x_{2,0}, x_{3,0}, x_{4,0}, x_{1,-1}, x_{2,-1}, x_{3,-1}\}, \\
E &:= \{\{F_{1,0}, x_{1,0}\}, \{F_{2,0}, x_{1,0}\}, \{F_{2,0}, x_{2,-1}\}, \{F_{2,0}, x_{1,0}\}, \{F_{3,0}, x_{1,0}\}, \{F_{3,0}, x_{2,0}\}, \\
&\qquad \{F_{3,0}, x_{3,-1}\}, \{F_{4,0}, x_{3,0}\}, \{F_{4,0}, x_{4,0}\}, \{F_{4,0}, x_{1,-1}\}\},
\end{aligned}
$$

where each $x_{i,j}$ denotes one equivalence class. We observe that there is no maximal matching in the shifting graph and we may choose either F_1 or F_2 as exposed (cf. Figure 3.5b). Since both are connected via a matching, we shift F_1 and F_2. Note, that F_3 has a path to F_1 and F_2 but the path is not an alternating path. Thus, we do not shift F_3. The equations F_1 and F_2 are only implicitly connected via the equivalence class $\{x_1, \dot{x}_1\}$. Constructing

Algorithm 3 Generalization of Algorithm 2 for shifting and differentiation

---

Input: graph $\mathtt{G} = (\mathtt{V_E} \dot\cup \mathtt{V_V}, \mathtt{E})$

denote by $\Delta_{\mathtt{m}\tau}\mathtt{F}_{\mathtt{j}}^{(\ell)}$, respectively $\Delta_{\mathtt{n}\tau}\mathtt{x}_{\mathtt{i}}^{(\mathtt{k})}$ the elements of $\mathtt{V_E}$, respectively $\mathtt{V_V}$

$\text{type} \in \{\mathtt{shift}, \mathtt{dif}\}$

graph $\mathtt{G^t} = (\mathtt{V_E^t} \dot\cup \mathtt{V_V^t}, \mathtt{E^t})$ of type $\mathtt{type}$

$$\mathtt{V_E^t} = \begin{cases} \left\{ \mathtt{F}_{\mathtt{j},\ell} \mid \Delta_{\mathtt{m}\tau}\mathtt{F}_{\mathtt{j}}^{(\ell)} \in \mathtt{V_E}, \ \mathtt{m}, \ell \in \mathbb{N}_0, \ \mathtt{j} \in \mathbb{N} \right\} & \text{if type} == \mathtt{dif} \\ \left\{ \mathtt{F}_{\mathtt{j},\ell} \mid \Delta_{\ell\tau}\mathtt{F}_{\mathtt{j}}^{(\mathtt{m})} \in \mathtt{V_E}, \ \mathtt{m}, \ell \in \mathbb{N}_0, \ \mathtt{j} \in \mathbb{N} \right\} & \text{if type} == \mathtt{shift} \end{cases}$$

$$\mathtt{V_V^t} = \begin{cases} \left\{ \mathtt{x}_{\mathtt{i},\mathtt{k}} \mid \Delta_{\mathtt{n}\tau}\mathtt{x}_{\mathtt{i}}^{(\mathtt{k})} \in \mathtt{V_V}, \ \mathtt{n}, \mathtt{k} \in \mathbb{N}_0, \ \mathtt{i} \in \mathbb{N} \right\} & \text{if type} == \mathtt{dif} \\ \left\{ \mathtt{x}_{\mathtt{i},\mathtt{k}} \mid \Delta_{\mathtt{k}\tau}\mathtt{x}_{\mathtt{i}}^{(\mathtt{n})} \in \mathtt{V_V}, \ \mathtt{k} \in \mathbb{Z}, \ \mathtt{n} \in \mathbb{N}_0, \ \mathtt{i} \in \mathbb{N} \right\} & \text{if type} == \mathtt{shift} \end{cases}$$

Output: graph $\mathtt{G} = (\mathtt{V_E} \dot\cup \mathtt{V_V}, \mathtt{E})$

1: $\mathtt{assign}(\mathtt{x}_{\mathtt{i},\mathtt{k}}) \leftarrow 0$ for each $\mathtt{x}_{\mathtt{i},\mathtt{k}} \in \mathtt{V_V^t}$
2: **for** $\mathtt{r} = 1, \dots, \mathtt{M}$ **do**
3:   $(\mathtt{p}, \mathtt{q}) \leftarrow (\mathtt{r}, 0)$
4:   **repeat**
5:    $\tilde{\mathtt{E}}^{\mathtt{t}} \leftarrow \mathtt{E^t} \setminus \{\{\mathtt{x}_{\mathtt{i},\mathtt{k}}, \mathtt{F}_{\mathtt{j},\ell}\} \in \mathtt{E^t} \mid \mathtt{x}_{\mathtt{i},\mathtt{k}+\mathtt{s}} \in \mathtt{V_V^t} \text{ for some } \mathtt{s} > 0\}$
6:    $\tilde{\mathtt{V}}_{\mathtt{V}}^{\mathtt{t}} \leftarrow \mathtt{V_V^t} \setminus \{\mathtt{x}_{\mathtt{i},\mathtt{k}} \mid \mathtt{x}_{\mathtt{i},\mathtt{k}+\mathtt{s}} \in \mathtt{V_V^t} \text{ for some } \mathtt{s} > 0\}$
7:    $\mathtt{colorV}(\mathtt{x}_{\mathtt{i},\mathtt{k}}) \leftarrow \mathtt{false}$ for each $\mathtt{x}_{\mathtt{i},\mathtt{k}} \in \tilde{\mathtt{V}}_{\mathtt{V}}^{\mathtt{t}}$
8:    $\mathtt{colorE}(\mathtt{F}_{\mathtt{j},\ell}) \leftarrow \mathtt{false}$ for each $\mathtt{F}_{\mathtt{j},\ell} \in \mathtt{V_E^t}$
9:    $(\mathtt{pathfound}, \mathtt{assign}, \mathtt{colorV}, \mathtt{colorE}) \leftarrow \text{Augmentpath}((\tilde{\mathtt{V}}_{\mathtt{E}}^{\mathtt{t}} \dot\cup \mathtt{V_V^t}, \tilde{\mathtt{E}}^{\mathtt{t}}), \mathtt{F}_{\mathtt{p},\mathtt{q}},$
    $\mathtt{assign}, \mathtt{colorV}, \mathtt{colorE})$
10:    **if** $\mathtt{pathfound} == \mathtt{false}$ **then**
11:     **if** $\text{type} == \mathtt{shift}$ **then**
12:      Differentiate equations according to Section 3.3
13:     **end if**
14:     $\mathtt{G^t} \leftarrow \text{updateGraph}(\mathtt{G^t}, \text{type}, \mathtt{boolFull} \leftarrow \mathtt{false}, \mathtt{colorV}, \mathtt{colorE})$
15:     $\mathtt{G} \leftarrow \text{updateGraph}(\mathtt{G}, \text{type}, \mathtt{boolFull} \leftarrow \mathtt{true}, \mathtt{colorV}, \mathtt{colorE})$
16:     **for each** $\mathtt{i}$ with $\mathtt{colorV}(\mathtt{x}_{\mathtt{i},\mathtt{k}}) == \mathtt{true}$ and $\mathtt{F}_{\mathtt{j},\ell} == \mathtt{assign}(\mathtt{x}_{\mathtt{i},\mathtt{k}})$ **do**
17:      $\mathtt{assign}(\mathtt{x}_{\mathtt{i},\mathtt{k}+1}) \leftarrow \mathtt{F}_{\mathtt{j},\ell}$
18:     **end for**
19:    **end if**
20:   **until** $\mathtt{pathfound} = \mathtt{true}$
21: **end for**

19

Algorithm 4 Pantelides Algorithm for DDAEs

Input: graph $\mathtt{G} = (\mathtt{V_E} \dot\cup \mathtt{V_V}, \mathtt{E})$, denote by $\Delta_{\mathtt{m}\tau}\mathtt{F}_{\mathtt{j}}^{(\ell)}$, respectively $\Delta_{\mathtt{n}\tau}\mathtt{x}_{\mathtt{i}}^{(\mathtt{k})}$ the elements of $\mathtt{V_E}$, respectively $\mathtt{V_V}$.

Output: graph $\mathtt{G}$

    Step 1: Shifting

1: $\mathtt{V_E^s} \leftarrow \left\{ \mathtt{F}_{\mathtt{j},0} \mid \mathtt{F}_{\mathtt{j}} \in \mathtt{V_E},\ \mathtt{j} \in \mathbb{N}_0 \right\}$

2: $\mathtt{V_V^s} \leftarrow \left\{ \mathtt{x}_{\mathtt{i},\mathtt{k}} \mid \Delta_{\mathtt{k}\tau}\mathtt{x}_{\mathtt{i}}^{(\mathtt{n})} \in \mathtt{V_V},\ \mathtt{k} \in \{-1, 0\},\ \mathtt{i} \in \mathbb{N},\ \mathtt{n} \in \{0, 1\} \right\}$

3: $\mathtt{E^s} \leftarrow \left\{ \{\mathtt{x}_{\mathtt{i},\mathtt{k}}, \mathtt{F}_{\mathtt{j},\ell}\} \mid \exists \mathtt{n}, \mathtt{m} \in \mathbb{N}_0,\ \mathtt{i}, \mathtt{j} \in \mathbb{N},\ \mathtt{k}, \ell \in \{-1, 0\} \text{ s.t. } \{\Delta_{\mathtt{k}\tau}\mathtt{x}_{\mathtt{i}}^{(\mathtt{n})}, \Delta_{\ell\tau}\mathtt{F}_{\mathtt{j}}^{(\mathtt{m})}\} \in \mathtt{E} \right\}$

4: $\mathtt{G^s} \leftarrow (\mathtt{V_V^s} \dot\cup \mathtt{V_E^s}, \mathtt{E^s})$

5: $\mathtt{G} \leftarrow \texttt{PantelidesDAE(G, shift, G^s)}$

    Step 2: Differentiation

6: $\mathtt{V_E^d} \leftarrow \left\{ \mathtt{F}_{\mathtt{j},\ell} \mid \Delta_{\mathtt{m}\tau}\mathtt{F}_{\mathtt{j}}^{(\ell)} \in \mathtt{V_E},\ \mathtt{m}, \ell \in \mathbb{N}_0,\ \mathtt{j} \in \mathbb{N} \right\}$

7: $\mathtt{V_V^d} \leftarrow \left\{ \mathtt{x}_{\mathtt{i},\mathtt{k}} \mid \Delta_{\mathtt{n}\tau}\mathtt{x}_{\mathtt{i}}^{(\mathtt{k})} \in \mathtt{V_V},\ \mathtt{n}, \mathtt{k} \in \mathbb{N}_0,\ \mathtt{i} \in \mathbb{N} \right\}$

8: $\mathtt{E^d} \leftarrow \left\{ \{\mathtt{x}_{\mathtt{i},\mathtt{k}}, \mathtt{F}_{\mathtt{j},\ell}\} \mid \exists \mathtt{n}, \mathtt{m}, \mathtt{k}, \ell \in \mathbb{N}_0,\ \mathtt{i}, \mathtt{j} \in \mathbb{N} \text{ s.t. } \{\Delta_{\mathtt{n}\tau}\mathtt{x}_{\mathtt{i}}^{(\mathtt{k})}, \Delta_{\mathtt{m}\tau}\mathtt{F}_{\mathtt{j}}^{(\ell)}\} \in \mathtt{E} \right\}$

9: $\mathtt{G^d} \leftarrow (\mathtt{V_E^d} \dot\cup \mathtt{V_V^d}, \mathtt{E^d})$

10: $\mathtt{G} \leftarrow \texttt{PantelidesDAE(G, dif, G^d)}$

---

the linear integer program as described in yields

$$\min \nu_1 + \nu_2 \quad \text{such that} \quad \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix} = \begin{bmatrix} 1 \end{bmatrix}, \quad \nu_1, \nu_2 \in \mathbb{N}_0$$

with the solution $\nu_1 = 1$, $\nu_2 = 0$. Consequently, we differentiate F_1. In the resulting graph Figure 3.5c a possible matching is giving by $M = \{\{\mathtt{F}_{1,1}, \mathtt{x}_{1,1}\}, \{\mathtt{F}_{2,1}, \mathtt{x}_{2,0}\}, \{\mathtt{F}_{4,0}, \mathtt{x}_{4,0}\}\}$. The matching is not maximal and the exposed vertex $\mathtt{F}_{3,1}$ is connected through an alternating path to $\mathtt{F}_{1,1}$ and $\mathtt{F}_{2,1}$. Thus we need to shift these three equations. The linear program for these three equations

$$\min \nu_1 + \nu_2 + \nu_3 \quad \text{such that} \quad \begin{bmatrix} 0 & 1 & -1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \nu_1, \nu_2, \nu_3 \in \mathbb{N}_0$$

with solution $\nu_1 = \nu_2 = 1$ and $\nu_3 = 0$ yields that we differentiate the first two equations. The resulting shifting graph Figure 3.5d provides a maximal matching. Now we construct the differentiation graph for the equations $\Delta_{2\tau}\ddot{F}_1, \Delta_{2\tau}\dot{F}_2, \Delta_{\tau}F_3$ and F_4 given by

$$V_E^s := \{\mathtt{F}_{1,2}, \mathtt{F}_{2,1}, \mathtt{F}_{3,0}, \mathtt{F}_{4,0}\},$$
$$V_V^s := \{\mathtt{x}_{1,0}, \mathtt{x}_{2,0}, \mathtt{x}_{3,0}, \mathtt{x}_{4,0}, \mathtt{x}_{1,1}, \mathtt{x}_{2,1}, \mathtt{x}_{3,1}, \mathtt{x}_{1,2}\},$$
$$E := \{\{\mathtt{F}_{1,2}, \mathtt{x}_{1,0}\}, \{\mathtt{F}_{1,2}, \mathtt{x}_{1,1}\}, \{\mathtt{F}_{1,2}, \mathtt{x}_{1,2}\}, \{\mathtt{F}_{2,1}, \mathtt{x}_{2,0}\}, \{\mathtt{F}_{2,1}, \mathtt{x}_{1,1}\}, \{\mathtt{F}_{2,1}, \mathtt{x}_{2,1}\},$$
$$\{\mathtt{F}_{2,1}, \mathtt{x}_{1,2}\}, \{\mathtt{F}_{3,0}, \mathtt{x}_{1,0}\}, \{\mathtt{F}_{3,0}, \mathtt{x}_{3,0}\}, \{\mathtt{F}_{3,0}, \mathtt{x}_{2,1}\}, \{\mathtt{F}_{4,0}, \mathtt{x}_{4,0}\}, \{\mathtt{F}_{4,0}, \mathtt{x}_{3,1}\}\},$$

or respectively Figure 3.6a. Note that the variable $\Delta_{-\tau}x_1$ does not occur in the graph since it is shifted. The vertex $\Delta_{\tau}F_3$ is exposed and connected through an alternating path to $\Delta_{2\tau}\ddot{F}_1$ and $\Delta_{2\tau}\dot{F}_2$. Thus we differentiate all three equations which results in Figure 3.6b. The existence of a maximal matching in Figure 3.6a leads to the termination of Algorithm 4. We conclude that we shift the first and second equation twice and that we differentiate the first equation three time, the second equation twice and the third equation once.   ◯
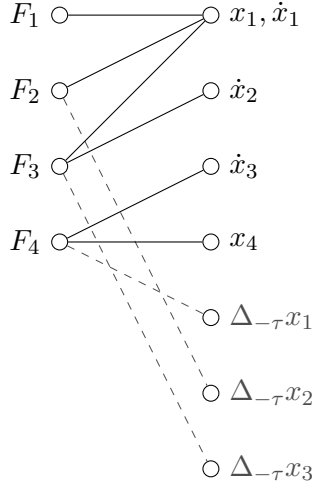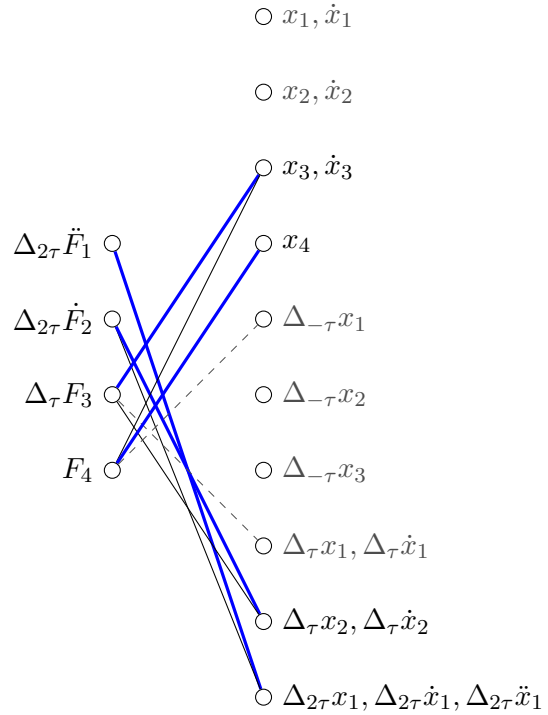
(a) Initial shifting graph
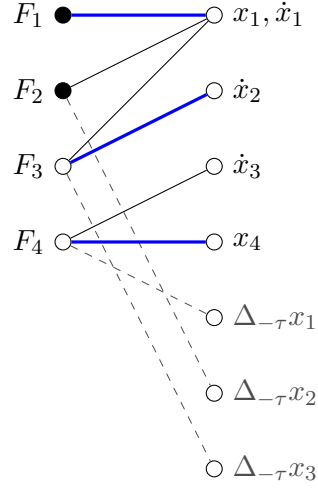
(b) Shifting graph with assignment
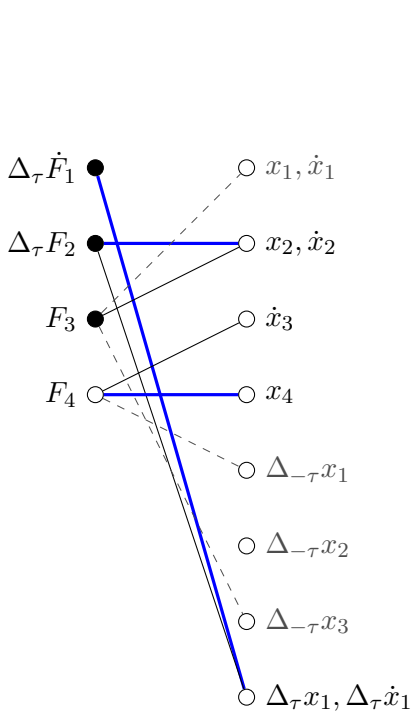
(c) Shifting graph after differentiating F_1 and shifting $\dot{F}_1$ and F_2 with assignment

(d) Shifting graph after differentiating $\Delta_\tau \dot{F}_1$, $\Delta_\tau F_2$ and shifting $\Delta_\tau \ddot{F}_1$, $\Delta_\tau \dot{F}_2$ and F_3

Figure 3.5: Visualization of Example 3.16 in the shifting step

---
Algorithm 5 UpdateGraph

Input: graph $G = (V_E \dot{\cup} V_V, E)$
$\qquad$ type $\in \{$shift, dif$\}$, boolFull $\in \{$true, false$\}$
$\qquad$ colorV, colorE
Output: $\quad$ graph $\tilde{G} = (\tilde{V}_V \dot{\cup} \tilde{V}_E, \tilde{E})$

1: **if** boolFull $==$ false **then**
2: $\qquad$ $x_{i,k,0} \leftarrow x_{i,k} \in V_V$ for all i, k
3: $\qquad$ $F_{j,\ell,0} \leftarrow F_{j,\ell} \in V_E$ for all j, ℓ
4: **else if** boolFull $==$ true and type $==$ shift **then**
5: $\qquad$ $x_{i,k,n} \leftarrow \Delta_{k\tau} x_i^{(n)} \in V_V$ for all i, k, n
6: $\qquad$ $F_{j,\ell,m} \leftarrow \Delta_{\ell\tau} F_j^{(m)} \in V_E$ for all j, ℓ, m
7: **else**
8: $\qquad$ $x_{i,k,n} \leftarrow \Delta_{n\tau} x_i^{(k)} \in V_V$ for all i, k, n
9: $\qquad$ $F_{j,\ell,m} \leftarrow \Delta_{m\tau} F_j^{(\ell)} \in V_E$ for all j, ℓ, m
10: **end if**
11: $\tilde{V}_V \leftarrow \{x_{i,k,n}\}$
12: $\tilde{V}_E \leftarrow \{F_{j,\ell,m}\}$
13: $\tilde{E} \leftarrow \{\{x_{i,k,n}, F_{j,\ell,m}\} \in E\}$
14: $\tilde{V}_V \leftarrow \tilde{V}_V \cup \{x_{i,k+1,n} \mid \text{colorV}(x_{i,k}) == \text{true}\}$
15: **for each** (j, ℓ) s.t. colorE$(F_{j,\ell}) ==$ true **do**
16: $\qquad$ $F_{j,\ell,m} \leftarrow F_{j,\ell+1,m}$
17: $\qquad$ **if** type $==$ shift **then**
18: $\qquad\qquad$ $\{x_{i,k,n}, F_{j,\ell,m}\} \in \tilde{E} \leftarrow \{x_{i,k+1,n}, F_{j,\ell,m}\}$ for each i, k, n
19: $\qquad$ **else**
20: $\qquad\qquad$ $\tilde{E} \leftarrow \tilde{E} \cup \{\{x_{i,k+1,n}, F_{j,\ell,m}\} \mid \{x_{i,k,n}, F_{j,\ell,m}\} \in \tilde{E}\}$
21: $\qquad$ **end if**
22: **end for**
23: $\tilde{G} \leftarrow (\tilde{V}_V \dot{\cup} \tilde{V}_E, \tilde{E})$
---

4 Termination of the new algorithm

Similarly as for the Pantelides algorithm we have to answer the question under which circumstances Algorithm 4 applied to the DDAE (1.1) terminates. As in the DAE case (cf. Theorem 2.10), we partition the set of variables Θ of (1.1) in such a way, that variables that only differ by the level of differentiation or shifting are grouped together. In more detail, we consider the equivalence relation

$$R_{\text{equal}} := \left\{ (\theta, \tilde{\theta}) \in \Theta \times \Theta \,\middle|\, \begin{array}{l} \text{there exist } k, \ell \in \mathbb{N},\, p, q \in \mathbb{N}_0 \text{ and } \xi \in \text{set}(x) \\ \text{such that } \theta = \Delta_{k\tau} \xi^{(p)},\, \tilde{\theta} = \Delta_{\ell\tau} \xi^{(q)} \end{array} \right\}. \quad (4.1)$$

It is easy to see that if Algorithm 4 applied to the DDAE (1.1) terminates, then each equation is matched to a different variable and thus (1.1) is structurally nonsingular with respect to Θ/R_{equal}. The following result shows that this is indeed also a sufficient condition.

Theorem 4.1. *Consider the DDAE (1.1) with set of variables Θ and equivalence relation R_{equal} as defined in (4.1). Algorithm 4 applied to (1.1) terminates if and only if (1.1) is structurally nonsingular with respect to Θ/R_{equal}.*
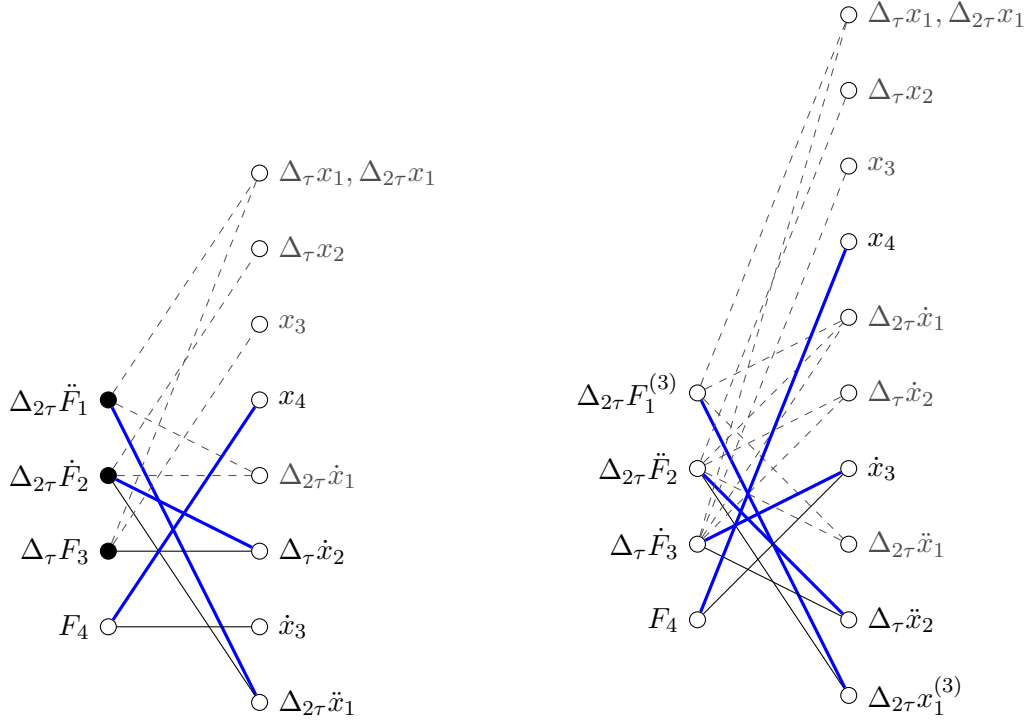
(a) Initial differentiation graph with assignment

(b) Differentiation graph after differentiating $\Delta_{2\tau}\ddot{F}_1$, $\Delta_{2\tau}\dot{F}_2$ and $\Delta_\tau F_3$

Figure 3.6: Visualization of Example 3.16 in the differentiation step

Since Algorithm 4 is divided into two parts, namely the shifting step and the differentiation step, we also split the proof of Theorem 4.1 in two parts. The shifting and differentiation step are very similar to the original Pantelides algorithm. Thus we can extend the termination proof presented in [23] to our setting. The main idea in [23] is to show that differentiating a MSS subset (see Definition 2.7) renders it structurally nonsingular and a structurally nonsingular subset remains structurally nonsingular after differentiation. Let us first show that under the assumptions of Theorem 4.1 the shifting step terminates.

Lemma 4.2. *Consider the DDAE* (1.1) *with set of variables* Θ *and equivalence relation* R_{equal} *as defined in* (4.1). *The shifting step of Algorithm 4 applied to* (1.1) *terminates if and only if* (1.1) *is structurally nonsingular with respect to* Θ/R_{equal}.

Proof. Recall that in the shifting step we group variables that are shifting similar, i.e., we work with the equivalence relation R_{s} from Definition 3.5. By virtue of Proposition 2.9, any subset of equations of the DDAE (1.1) that Algorithm 4, or more precisely Algorithm 3, identifies to be shifted is MSS with respect to Θ/R_{s}. Suppose that the subset

$$\widehat{F}(t, \tilde{x}, \dot{\overline{x}}, \Delta_{-\tau}\widehat{x}) = 0, \tag{4.2}$$

with $\text{set}(\tilde{x}, \overline{x}, \widehat{x}) \subseteq \text{set}(x)$ of the DDAE (1.1) is such a set. Note that here the sets

$$\text{set}(\tilde{x}), \ \text{set}(\overline{x}), \ \text{and} \ \text{set}(\widehat{x})$$

are not necessarily disjoint. More precisely, decompose $\widehat{x} = (\widehat{x}_1, \widehat{x}_2)$ such that

$$\text{set}(\widehat{x}) = \text{set}(\widehat{x}_1)\dot{\cup}\,\text{set}(\widehat{x}_2), \qquad \text{set}(\widehat{x}_2) \cap \text{set}(\overline{x}, \tilde{x}) = \emptyset, \qquad \text{and} \qquad \text{set}(\widehat{x}_1) \subseteq \text{set}(\overline{x}, \tilde{x}).$$

23

Let us assume first that the DDAE (1.1) is structurally nonsingular with respect to Θ/R_{equal}. This implies that $\text{set}(\widehat{x}_2) \neq \emptyset$. Shifting (4.2) results in the equation $\Delta_\tau \widehat{F}$ given by

$$\widehat{F}(t + \tau, \Delta_\tau \tilde{x}, \Delta_\tau \dot{\tilde{x}}, \widehat{x}) = 0.$$

By construction we have

$$\left| \text{set}(\Delta_\tau \widehat{F}) \right| = \left| \text{set}(\widehat{F}) \right| \leq \left| \text{set}\left(\tilde{x}, \dot{\tilde{x}}, \Delta_{-\tau} \widehat{x}\right) / R_{\text{equal}} \right| = \left| \text{set}\left(\tilde{x}, \dot{\tilde{x}}, \Delta_{-\tau} \widehat{x}_2\right) / R_{\text{equal}} \right|$$
$$= \left| \text{set}\left(\tilde{x}, \dot{\tilde{x}}, \Delta_{-\tau} \widehat{x}_2\right) / R_{\text{shift}} \right| = \left| \text{set}\left(\Delta_\tau \tilde{x}, \Delta_\tau \dot{\tilde{x}}, \widehat{x}_2\right) / R_{\text{shift}} \right|$$

showing that $\Delta_\tau \widehat{F}$ is structurally nonsingular with respect to the equivalence relation R_{shift}. Note that this argument is easily extended to subsets of the general equation (3.4) and we conclude that whenever a MSS subset of equations is shifted during the shifting step it becomes structurally nonsingular. In Algorithm 3 we do not only replace $\widehat{F}$ with $\Delta_\tau \widehat{F}$ but may have to differentiate some of the equations. However, Proposition 3.8 implies that this does not affect the shifting graph and hence does not effect the structural nonsingularity. In order to show that the shifting step terminates it suffices to show that any subset $\text{set}(\overline{F}) \subseteq \text{set}(F)$ that is disjoint from $\text{set}(\widehat{F})$ and is structurally nonsingular before we shift (4.2) remains structurally nonsingular after shifting. The only possibility for $\text{set}(\overline{F})$ to become structurally singular is if a subset of $\text{set}(\overline{F})$ has a matching edge to some vertex in $\text{set}(\tilde{x}, \dot{\tilde{x}})/R_s$. This implies that there is an alternating path from the exposed equation in $\widehat{F}$ to one of the equations in $\text{set}(\overline{F})$, a contradiction to $\text{set}(\widehat{F}) \cap \text{set}(\overline{F}) = \emptyset$. We conclude that the shifting step terminates.

Conversely, assume that (1.1) is structurally singular with respect to Θ/R_{equal}. Then at some point Algorithm 4 identifies a subset (4.2) of (1.1) that is structurally singular with respect to Θ/R_{equal}. In this case, we have $\text{set}(\widehat{x}_2) = \emptyset$ and thus

$$\left| \text{set}(\widehat{F}) \right| > \left| \text{set}\left(\tilde{x}, \dot{\tilde{x}}, \Delta_{-\tau} \widehat{x}\right) / R_{\text{equal}} \right| = \left| \text{set}\left(\tilde{x}, \dot{\tilde{x}}, \Delta_{-\tau} \widehat{x}_2\right) / R_{\text{shift}} \right| \geq \left| \text{set}\left(\tilde{x}, \dot{\tilde{x}}\right) / R_{\text{shift}} \right|.$$

In this case, shifting does not increase the set of equivalence classes and thus the set (4.2) is still structurally singular with respect to the relation R_{shift} and thus results in an infinite loop. $\qquad\square$

To show that also the differentiation step in Algorithm 4 terminates, we want to apply Theorem 2.10. For this we have to show that the set of equations that we obtain after applying the shifting step is structurally nonsingular with respect to $\widetilde{\Theta}/R_{\text{equal}}$ with

$$\widetilde{\Theta} := \{\theta \in \Theta \mid (\theta, \theta) \in R_{\text{diff}}\}. \tag{4.3}$$

Note that $\widetilde{\Theta}$ does not include variables depending on $t - \tau$ and hence $\widetilde{\Theta}/R_{\text{equal}}$ is different from Θ/R_{equal}.

Lemma 4.3. *Suppose that* (1.1) *is structurally nonsingular with respect to* Θ/R_{equal}. *Let* $G = (V_E \dot\cup V_V, E)$ *with*

$$V_E = \left\{\Delta_{m_1\tau} F_1^{(\ell_1)}, \ldots, \Delta_{m_M\tau} F_M^{(\ell_M)}\right\}.$$

denote the graph that results from applying the shifting step of Algorithm 4 to (1.1). *Define the set of equations*

$$0 = \overline{F}(\bar{t}, \overline{x}, \dot{\overline{x}}, \Delta_{-\tau} x) = [\overline{F}_i]_{i=1,\ldots,M}(\bar{t}, \overline{x}, \dot{\overline{x}}, \Delta_{-\tau} x) \tag{4.4}$$

with $m := \max_i(m_i)$, $\overline{\xi} = \begin{bmatrix} \Delta_{0\tau}\xi & \dots & \Delta_{m\tau}\xi \end{bmatrix}$ *for* $\xi \in \{t, x\}$ *by* $\overline{F}_i := \Delta_{m_i} F_i$. *Then* (4.4) *is structurally nonsingular with respect to* $\widetilde{\Theta}/R_{\text{equal}}$ *with* $\widetilde{\Theta}$ *defined in* (4.3).

Proof. Since (1.1) is structurally nonsingular with respect to Θ/R_{equal} Algorithm 4 up to Line 5 terminates (cf. Lemma 4.2). Thus, we can assign each equation $\overline{F}_i$ for $i = 1, \dots M$ to a highest shift. In other words, there exists $k \in \{0, \dots, m\}^{|\text{set}(x)|}$ such that

$$|\text{set}(\overline{F})| = |\text{set}(\Delta_{k\tau}\dot{x})| = \left|\widetilde{\Theta}/R_{\text{equal}}\right|$$

holds which is the structural nonsingularity of $\overline{F}$ with respect to $\widetilde{\Theta}/R_{\text{equal}}$. $\qquad\square$

Combining the previous results, we are now able to proof Theorem 4.1.

Proof of Theorem 4.1. If (1.1) is structurally nonsingular with respect to Θ/R_{equal} Algorithm 4 up to Line 5 terminates (cf. Lemma 4.2). Since the resulting shifted equations (4.4) are structurally nonsingular with respect to $\widetilde{\Theta}/R_{\text{equal}}$, we can use Theorem 2.10 to conclude that Algorithm 4 terminates. The converse direction follows from Lemma 4.2. $\quad\square$

We conclude our analysis by investigating in which cases Algorithm 4 determines that no equation needs to be shifted. The following result shows that this is the case if the DDAE (1.1) is structurally nonsingular with respect to the set of equivalence classes that are obtained by using the relation R_{equal} with a restricted set of variables.

Theorem 4.4. *If the set of equations* (1.1) *is structurally nonsingular with respect to* $\overline{\Theta}/R_{\text{equal}}$ *with*

$$\overline{\Theta} := \left\{\xi \in \text{set}\,(x, \dot{x}) \,\middle|\, \frac{\partial F}{\partial \xi} \not\equiv 0\right\}$$

then Algorithm 4 applied to (1.1) *terminates and determines that no equation is shifted. In this case also Algorithm 2 applied to* (1.1) *where we replace* $\Delta_{-\tau}x$ *with a function parameter* λ *terminates and the resulting graphs of both algorithms are isomorphic.*

Proof. First observe that the structural nonsingularity of (1.1) with respect to $\overline{\Theta}/R_{\text{equal}}$ implies structural nonsingularity with respect to Θ/R_{equal}. Thus Theorem 4.1 ensures that Algorithm 4 terminates if applied to (1.1). Consider a subset

$$\widehat{F}(t, \eta, \gamma, \dot{\gamma}, \dot{\mu}, \Delta_{-\tau}\widehat{x}) = 0$$

of the DDAE (1.1) with $\text{set}(\eta)\dot{\cup}\text{set}(\gamma)\dot{\cup}\text{set}(\mu) \subseteq \text{set}(x)$ and $\text{set}(\widehat{x}) \subseteq \text{set}(x)$. The structural nonsingularity with respect to $\overline{\Theta}/R_{\text{equal}}$ implies

$$\left|\text{set}(\widehat{F})\right| \leq |\text{set}\,(\eta, \gamma, \dot{\gamma}, \dot{\mu})\,/R_{\text{equal}}| = |\text{set}\,(\eta, \gamma, \dot{\gamma}, \dot{\mu})\,/R_{\text{shift}}|$$

which is the structural nonsingularity of $\widehat{F}$ after deleting variables with lower shifts in the shifting graph. Thus no equation is shifted. We conclude that Algorithm 4 can be reduced to the differentiation step. Within the differentiation step, Algorithm 2 and Algorithm 3 coincide and thus Algorithm 2 terminates with the same result as Algorithm 4. $\quad\square$

We conclude our analysis by emphasizing that the Pantelides algorithm for DDAEs suffers from the same problems as the original Pantelides algorithm (see for instance Example 2.11 and the discussion thereafter). In particular, there is no guarantee that the correct number of differentiations and shifts is identified.

Example 4.5. Even though the solution of the DDAE

$$0 = x_4 + \Delta_{-\tau} x_1 + f_1, \quad \dot{x}_2 = x_1 + f_1, \quad 0 = x_2 + x_4 + f_3, \quad 0 = x_2 + \Delta_{-\tau} x_3 + f_4$$

depends on the shifted and differentiated equations F_1, F_3, F_4, the Pantelides algorithm for DDAEs determines that we have to shift equations F_1, F_3, and F_4 but do not have to differentiate any equation. ◯

5 Summary

We have presented a method (Algorithm 4) to determine which equations of the DDAE (1.1) need to be shifted and which equations need to be differentiated. The algorithm extends the Pantelides algorithm [23] for DAEs to the DDAE case. The main idea that enables the generalization is the introduction of equivalence classes in the bipartite graph associated with the DDAE (1.1). For further details see Definition 2.1. The Pantelides algorithm for DDAEs is divided into the shifting step and the differentiation step. We prove (Proposition 3.8) that differentiation does not affect the shifting graph and hence start with the shifting step. It turns out that already in the shifting step we have to differentiate some equations. To obtain the required number of differentiations for each equation, we may have to solve a sequence of linear systems, see Section 3.3 for further details. We presented a necessary and sufficient condition for the termination of Algorithm 4 in Theorem 4.1. We foresee that we can combine our framework with a dummy derivative approach [21] and an algebraic approach [28] such that Algorithm 4 can be used with numerical time integration methods.

Since our framework builds upon the Pantelides algorithm, it has similar strengths and weaknesses. One of the big advantages of our algorithm (in contrast to the algebraic index reduction procedure in [14]) is that the underlying graph-theoretical tools do not suffer from numerical rank decisions and can be computed efficiently also in a large-scale setting. On the other hand, our algorithm is not invariant under equivalence transformations and thus may fail to determine the correct number of shifts and differentiations. In the DAE case, this fact is accounted for in the Σ-method [25, 26] by including a success check. It is ongoing research to develop such a success check for DDAEs.

References

[1] U. M. Ascher and L. R. Petzold. The numerical solution of delay-differential-algebraic equations of retarded and neutral type. *SIAM J. Numer. Anal.*, 32(5):1635–1657, 1995.

[2] A. Bellen and M. Zennaro. *Numerical methods for delay differential equations.* Clarendon Press, 2003.

[3] R. Bellman and K. Cooke. *Differential-difference equations.* Academic Press, New York, 1963.

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

[5] S. L. Campbell. A general form for solvable linear time varying singular systems of differential equations. *SIAM J. Math. Anal.*, 18(4):1101–1115, 1987.

[6] S. L. Campbell. Nonregular 2D descriptor delay systems. *IMA Journal of Mathematical Control and Information*, 12(1):57–67, 1995.

[7] I. S. Duff. On algorithms for obtaining a maximum transversal. *ACM Trans. Math. Softw.*, 7(3):315–330, 1981.

[8] T. Erneux. *Applied Delay Differential Equations*, volume 4. Springer-Verlag, New York, 2009.

[9] E. Fosong, P. Schulze, and B. Unger. From time-domain data to low-dimensional structured models. *ArXiv e-print 1902.05112*, Feb 2019.

[10] F. R. Gantmacher. *The Theory of Matrices*, volume 2. Chelsea Publishing Company, New York, NY, USA, 1959.

[11] H. Gluesing-Luerssen. *Linear Delay-Differential Systems with Commensurate Delays: An Algebraic Approach*. Springer, Berlin, Berlin, 2002.

[12] P. Ha. *Analysis and Numerical solutions of Delay Differential-Algebraic Equations*. Dissertation, Technische Universität Berlin, 2015.

[13] P. Ha and V. Mehrmann. Analysis and reformulation of linear delay differential-algebraic equations. *Electron. J. Linear Algebr.*, 23:703–730, 2012.

[14] P. Ha and V. Mehrmann. Analysis and numerical solution of linear delay differential-algebraic equations. *BIT Numer Math*, 56(2):633–657, 2016.

[15] P. Ha, V. Mehrmann, and A. Steinbrecher. Analysis of linear variable coefficient delay differential-algebraic equations. *J. Dynam. Differential Equations*, 26(4):889–914, 2014.

[16] J. K. Hale and S. M. Verduyn Lunel. *Introduction to Functional Differential Equations*. Springer-Verlag, New York, 1993.

[17] D. Jungnickel. *Graphs, networks and algorithms*. Springer, Berlin, 4 edition, 2013.

[18] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[19] P. Kunkel and V. Mehrmann. Regular solutions of nonlinear differential-algebraic equations and their numerical determination. *Numer. Math.*, 79(4):581–600, 1998.

[20] P. Kunkel and V. Mehrmann. *Differential-Algebraic Equations. Analysis and Numerical Solution*. European Mathematical Society, Zürich, Switzerland, 2006.

[21] S. E. Mattsson and G. Söderlind. Index reduction in differential-algebraic equations using dummy derivatives. *SIAM J. Sci. Comput.*, 14(3):677–692, 1993.

[22] J. A. Nelder and R. L. Mead. A simplex method for function minimization. *Comput. J.*, 7:308–313, 1965.

[23] C. C. Pantelides. The consistent initialization of differential-algebraic systems. *SIAM J. ScI. STAT. Comput.*, 9(2):213–231, 1988.

[24] L. Petzold. Differential/algebraic equations are not ODE's. *SIAM J. ScI. STAT. Comput.*, (3):367–384.

[25] J. D. Pryce. A simple structural analysis method for DAEs. *BIT Numer. Math.*, 41(2):364–394, 2001.

[26] J. D. Pryce, N. S. Nedialkov, and G. Tan. DAESA—a matlab tool for structural analysis of differential-algebraic equations: Theory. *ACM Transactions on Mathematical Software*, 41(2):9:1–9:20, 2015.

[27] G. Reissig, W. S. Martinson, and P. I. Barton. Differential–Algebraic Equations of Index 1 May Have an Arbitrarily High Structural Index. *SIAM J. Sci. Comput.*, 21(6):1987–1990, 2000.

[28] L. Scholz and A. Steinbrecher. Regularization of DAEs based on the signature method. *BIT Numerical Mathematics*, 56:319–340, 2016.

[29] L. Scholz and A. Steinbrecher. Structural-algebraic regularization for coupled systems of DAEs. *BIT Numerical Mathematics*, 56(2):777–804, 2016.

[30] P. Schulze and B. Unger. Data-driven interpolation of dynamical systems with delay. *Systems Control Lett.*, 97:125–131, 2016.

[31] P. Schulze, B. Unger, C. Beattie, and S. Gugercin. Data-driven structured realization. *Linear Algebra Appl.*, 537:250 – 286, 2018.

[32] S. Trenn and B. Unger. Delay regularity of differential-algebraic equations. Preprint, submitted for publication, 2019. Available from https://stephantrenn.net/wp-content/uploads/2019/03/Preprint-TU190317.pdf.

[33] B. Unger. Discontinuity propagation in delay differential-algebraic equations. *Electron. J. Linear Algebr.*, 34:582–601, 2018.

[34] D. B. West. *Introduction to Graph Theory.* Prentice Hall, Upper Saddle River, 2 edition, 2001.

List of Abbreviations

DAE	differential-algebraic equation
DDAE	delay differential-algebraic equation
DDE	delay differential equation
MSS	minimal structural singular
ODE	ordinary differential equation