BIT Numer Math (2017) 57:433–462
DOI 10.1007/s10543-016-0642-9

Structural analysis based dummy derivative selection for differential algebraic equations

Ross McKenzie[1] · **John Pryce**[1]

Received: 31 March 2016 / Accepted: 8 November 2016 / Published online: 17 February 2017

Abstract The signature matrix structural analysis method developed by Pryce provides more structural information than the commonly used Pantelides method and applies to differential-algebraic equations (DAEs) of arbitrary order. It is useful to consider how existing methods using the Pantelides algorithm can benefit from such structural analysis. The dummy derivative method is a technique commonly used to solve DAEs that can benefit from such exploitation of underlying DAE structures and information found in the Signature Matrix method. This paper gives a technique to find structurally necessary dummy derivatives and how to use different block triangular forms effectively when performing the dummy derivative method and then provides a brief complexity analysis of the proposed approach. We finish by outlining an approach that can simplify the task of dummy pivoting.

Keywords DAEs · Numerical Analysis

Mathematics Subject Classification 34A09 · 65L80

1 Introduction

Differential-algebraic equations (DAEs) arise from the equation based modelling of physical systems, such as those found in engineering or physics, with problems arising,

Communicated by Anne Kværnø.

✉ Ross McKenzie
mckenzier1@cardiff.ac.uk

John Pryce
prycejd1@cardiff.ac.uk

[1] Cardiff University, Cardiff, Wales

for instance, from chemical engineering [21], electronic circuits [1] and robotics [2]. Models are now frequently built interactively using different components from large libraries in environments such as gPROMS and Simulink, as well as MapleSim and other tools that use the Modelica language. This way of modelling can lead to large DAEs [4]. A common notion in DAE theory is the *differentiation index*—which is equal to the number of times all or part of the system has to be differentiated in order to solve the problem as an ODE. It is well known that solving a high index (larger than one) DAE directly is numerically difficult, hence modelling tools usually perform some structural analysis (SA) to determine the index of the problem. This analysis is usually based on Pantelides' algorithm [14], although we will be using the (equivalent for DAEs of order 1) Signature Matrix method [18] for the duration of this paper, as it applies to DAEs of arbitrary order and provides us with extra structural information we wish to exploit. As such if we talk about structural analysis of a DAE we mean information obtained by applying the Signature Matrix method to that DAE. After finding a DAE's index, different packages handle high index DAEs very differently, for example gPROMS [16] reports a set of equations and variables that are causing the high index, whereas DYMOLA [3] performs an index reduction algorithm (specifically dummy derivatives [8]) to form a new equivalent index 1 DAE, which it then solves. We are interested in the latter approach. We shall consider a DAE given by:

$$f_i(t, \text{ the } x_j \text{ and derivatives of them}), \quad i = 1, \ldots, n$$

where $x_j(t)$, $j = 1, \ldots, n$ are state variables and functions of some independent variable t, usually considered to be time. DAEs we consider can be fully non-linear but must be solvable, i.e. have a unique smooth solution when given a sufficient set of consistent initial conditions. The functions f_i must be least smooth enough to find derivatives required by the method to succeed, usually three times differentiable is sufficient in practical examples.

2 A review of the signature matrix method

We begin by providing the reader with a needed background in the Signature Matrix method, for a more detailed explanation see [18] and [20].

2.1 Basic Structural information

Take a DAE as above, we begin by finding the problem's *Signature Matrix* Σ, with entries:

$$\sigma_{i,j} = \begin{cases} \text{order of highest derivative of } x_j \text{ in } f_i & \text{if } x_j \text{ occurs in } f_i \\ -\infty & \text{if not.} \end{cases}$$

Then find a *highest value transversal* (HVT). A transversal, T, is any choice of n positions (i, j) in an $n \times n$ matrix, such that only one entry in each row and column is

selected. If we consider a transversal T, then we say $\mathrm{Val}(T) = \sum_{(i,j)\in T} \sigma_{i,j}$. A HVT, say $\mathcal{T}$, is such that $\mathrm{Val}(\mathcal{T}) \geq \mathrm{Val}(T)$ for all transversals T—we denote Val $(\mathcal{T})$ as $\mathrm{Val}(\Sigma)$. We call a DAE *structurally well posed* if $\mathrm{Val}(\Sigma)$is finite, i.e. there exists a HVT with corresponding entries in Σ all finite. If the DAE is structurally well posed (SWP) and solvable, see [13], $\mathrm{Val}(\Sigma)$is equal to its *degrees of freedom* (DOF). In this paper we only consider SWP DAEs.

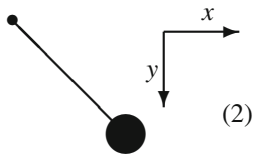
We then find non-negative integer valued vectors, **c** and **d** satisfying:

$$\sigma_{i,j} \leq d_j - c_i \qquad \text{with equality on a HVT} \tag{1}$$

called *valid offset vectors*. There exists a unique elementwise smallest choice of **c** and **d** termed the *canonical offset vectors*. It is easily seen that such canonical offsets must have $\min_i c_i = 0$. Unless otherwise stated we use canonical offsets throughout the paper. Finding a HVT and offsets is a *linear assignment problem* [18].

Example 1 Consider the simple pendulum:

$$\left.\begin{aligned} A(t) &= \ddot{x}(t) + \lambda(t)x(t) &&= 0 \\ B(t) &= \ddot{y}(t) + \lambda(t)y(t) - G &&= 0 \\ C(t) &= x^2(t) + y^2(t) - L^2 &&= 0 \end{aligned}\right\} \tag{2}$$



Pendulum bob, mass=1

Equations A and B are from Newton's second law, since we are measuring to the right in the x direction and downwards in the y direction. We keep the length of the pendulum fixed, which is how we get equation C. Displacement in the horizontal and vertical directions is measured by x and y respectively, λ is a multiple of tension in the rod. Gravity and rod length are given by G and L respectively. This is an index 3 formulation of the pendulum, if we differentiate equation c twice we can solve for $\ddot{x}$ and $\ddot{y}$ but λ is undifferentiated, so we must differentiate each equation once more, hence the problem is index 3. One can see from this simple example that it's easy to construct a high index DAE. One may think this problem is rather artificial, since changing to polar coordinates yields an ODE, however for large scale models found in practice such a manipulation is usually not obvious.

We form the problem's signature matrix as above, with canonical offsets around the side:

$$\Sigma = \begin{array}{c} \\ A \\ B \\ C \\ d_j \end{array} \begin{array}{c} \begin{array}{ccc} x & y & \lambda \end{array} \\ \begin{pmatrix} 2^{\bullet} & -\infty & 0^{\circ} \\ -\infty & 2^{\circ} & 0^{\bullet} \\ 0^{\circ} & 0^{\bullet} & -\infty \end{pmatrix} \\ \begin{array}{ccc} 2 & 2 & 0 \end{array} \end{array} \begin{array}{c} c_i \\ 0 \\ 0 \\ 2 \\ \end{array}$$

There are two HVTs, marked by $\bullet$ and $\circ$ in the signature matrix above. Comparing the offsets of the problem with what made the problem index 3 we see we have a scheme of which equations to differentiate and what their highest order variables will be. We go in to more detail on the ordering of this scheme in Sect. 2.2.

2.2 Structural analysis stages

The example above leads us to the concept of a *structural index* ν_s, defined as:

$$\nu_S = \max_i c_i + \begin{cases} 1 & \text{if any } d_j \text{ is zero,} \\ 0 & \text{if not.} \end{cases}$$

The $+1$ if a $d_j = 0$ case is needed to add a first order derivative of otherwise undifferentiated variables, see λ in Example 1. If the DAE is SWP ν_S is known to be an upper bound on the differentiation index, and in many applications they are equal, which is why we suggest using it as a basis for an index reduction algorithm such as DDs later. We now differentiate each equation in stages specified by **c**. Initially the approach given by the Signature Matrix method was to solve parts of the DAE in stages via Taylor series, see [12], however we just focus on the equations used at each stage and the variables being solved for.

We briefly inform the reader of how we denote derivatives. Consider for instance variables x and y, both of which depend on t, and some function f which is a function of t, x and y and perhaps some first order derivatives of x, i.e. $f = f(x, \dot{x}, y, t)$. Then:

$$\dot{f} = \frac{\partial f}{\partial x}\dot{x} + \frac{\partial f}{\partial \dot{x}}\ddot{x} + \frac{\partial f}{\partial y}\dot{y} + \frac{\partial f}{\partial t}.$$

This example shows we mean ordinary derivative with respect to t unless otherwise stated.

The offsets of a DAE provide us with a staged solution process starting at stage $k = -\max_j d_j = k_{\min}$ and increasing by one at subsequent stages where at each stage we use equations

$$f_i^{(k+c_i)} \quad \forall i \text{ such that } k + c_i \geq 0 \tag{3}$$

to solve for variables

$$x_j^{(k+d_j)} \quad \forall j \text{ such that } k + d_j \geq 0. \tag{4}$$

Let m_k be the number of i such that $k + c_i \geq 0$ and n_k be the number of j such that $k + d_j \geq 0$, so that at stage k we solve m_k equations for n_k unknowns. Note: for $k < -\max_j d_j$ we have $m_k = n_k = 0$ and for $k \geq 0$ we have that $m_k = n_k = n$.

Example 2 Consider again the simple pendulum (2), it has SA stages as given in Table 1.

We now make use of the following Lemma, found in [5].

Lemma 1 (Griewank's Lemma) *Suppose $f(x_1, x_2, \ldots, x_n)$ is a smooth function that contains derivatives of x_j not exceeding order m_j. Then the $(m_j + 1)$st derivative of x_j occurs linearly in $\dot{f} = \frac{df}{dt}$, and*

$$\frac{\partial \dot{f}}{\partial x_j^{(m_j+1)}} = \frac{\partial f}{\partial x_j^{(m_j)}}.$$

Table 1 Stages for the simple pendulum

k	Equations being used	Variables being found	Using variables	m_k	n_k
-2	C	x, y	N/A	1	2
-1	$\dot{C}$	$\dot{x}, \dot{y}$	x, y	1	2
0	$A, B, \dot{C}$	$\ddot{x}, \ddot{y}, \lambda$	$\dot{x}, \dot{y}\, x, y$	3	3
1	$\dot{A}, \dot{B}, C^{(3)}$	$x^{(3)}, y^{(3)}, \dot{\lambda}$	$\ddot{x}, \ddot{y}, \lambda\, \dot{x}, \dot{y}\, x, y$	3	3
...	...	...	...	3	3

We have the following formula for the System Jacobian $\mathbf{J}$, with first equality coming from Lemma 1—as shown in [18]:

$$\mathbf{J}_{i,j}(c_i, d_j) = \frac{\partial f_i^{(c_i)}}{\partial x_j^{(d_j)}} = \frac{\partial f_i}{\partial x_j^{(d_j - c_i)}} = \begin{cases} \partial f_i / \partial x_j^{(\sigma_{i,j})} & \text{if } d_j - c_i = \sigma_{i,j} \\ 0 & \text{elsewhere.} \end{cases} \tag{5}$$

Note: $\mathbf{J}$ depends on the choice of offsets, unless otherwise stated by the context we use canonical offsets. If there exists a consistent point at which $\mathbf{J}$ is non singular then the method succeeds and the DAE is locally solvable. Each stage k of SA listed above uses an $m_k \times n_k$ submatrix of $\mathbf{J}$, denoted by $\mathbf{J}_k$, containing the rows and columns for equations and variables used at stage k from (3) and (4).

Example 3 Consider again the simple pendulum. Let $\mathbf{J}_{<i}$ be taken to mean all Jacobians used in stages $i-1, i-2, \ldots$ and similarly for $\mathbf{J}_{>i}$, we have the following stage wise system Jacobians for the simple pendulum, where [ ] denotes an empty matrix:

$$\mathbf{J} = \mathbf{J}_0 = \begin{array}{c} \\ A \\ B \\ \ddot{C} \end{array} \begin{array}{c} \begin{array}{ccc} \ddot{x} & \ddot{y} & \lambda \end{array} \\ \begin{pmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 2x & 2y & 0 \end{pmatrix} \end{array},$$

$$\mathbf{J}_{-1} = \begin{array}{c} \\ \dot{C} \end{array} \begin{array}{c} \begin{array}{cc} \dot{x} & \dot{y} \end{array} \\ \begin{pmatrix} 2x & 2y \end{pmatrix} \end{array}, \quad \mathbf{J}_{-2} = \begin{array}{c} \\ C \end{array} \begin{array}{c} \begin{array}{cc} x & y \end{array} \\ \begin{pmatrix} 2x & 2y \end{pmatrix} \end{array}, \quad \mathbf{J}_{<-2} = [\,]$$

2.3 SA based Block Triangular Forms

We discuss block forms natural to the Signature method, for more detail see [19].

A natural sparsity pattern for a DAE is the set where the entries of Σ are finite:

$$S = \{(i, j) \mid \sigma_{i,j} > -\infty\} \qquad \text{(the sparsity pattern of } \Sigma). \tag{6}$$

A more informative BTF comes from the sparsity pattern of the system Jacobian $\mathbf{J}$:

$$S_0 = S_0(c, d) = \{(i, j) \mid d_j - c_i = \sigma_{i,j}\} \qquad \text{(the sparsity pattern of } \mathbf{J}). \tag{7}$$

$$S_0(c, d) \subseteq S \quad \text{for any } c, d.$$

In applications a Block Triangular Form (BTF) based on S_0 is usually significantly finer than one based on S and as such we call a BTF based on S the coarse BTF and on S_0 the fine BTF. We now define the concept of a local offset:

Definition 1 (*Local offsets*) We denote canonical local offsets of the fine BTF as $\hat{\mathbf{c}}$ and $\hat{\mathbf{d}}$, that is the offsets found by treating each fine block as a stand alone DAE.

We have following theorem from [20]:

Theorem 1 *The difference between local and global offsets is a constant over a fine block*

We call the difference between local and global offsets the lead time of a block l and denote it K_l.

Example 4 Consider a modified double pendulum DAE:

$$\left.\begin{aligned} f_1 &= \ddot{x}_1 + x_1 x_3 && =0 \\ f_2 &= \ddot{x}_2 + x_2 x_3 - G && =0 \\ f_3 &= x_1^2 + x_2^2 - L^2 && =0 \\ f_4 &= \ddot{x}_4 + x_4 x_6 && =0 \\ f_5 &= (x_5^{(3)})^2 + x_5 x_6 - G && =0 \\ f_6 &= x_4^2 + x_5^2 - (L + c x_3)^2 + \ddot{x}_3 && =0 \end{aligned}\right\} \tag{8}$$

We present the different BTFs (produced by DAESA) indicated by dotted lines in Figs. 1, 2 and 3, where a blank means $-\infty$.

3 A review of the dummy derivative method

We provide the reader with a background in the dummy derivative (DD) method, for a more detailed explanation we defer to [7,8].

3.1 The idea behind DDs

The usual manner of solving a DAE involves differentiating parts of the DAE in order to reveal the so called *hidden constraints* of the problem, as demonstrated above in the pendulum example. The DD method takes an $n \times n$ smooth DAE and adds the hidden constraint equations to the system directly, as opposed to solving them stage-wise as above. The issue with this is that the system is now over-determined. To get a square system again the DD method adds extra dependent variables to the system for each constraint equation, specifically the method finds a subset of appearing derivatives of variables to be considered algebraic. It is shown in [8] that such a formulation has the same solution as the original DAE. The advantage of this approach is by solving the resulting index 1 DAE we satisfy all constraint equations of the original

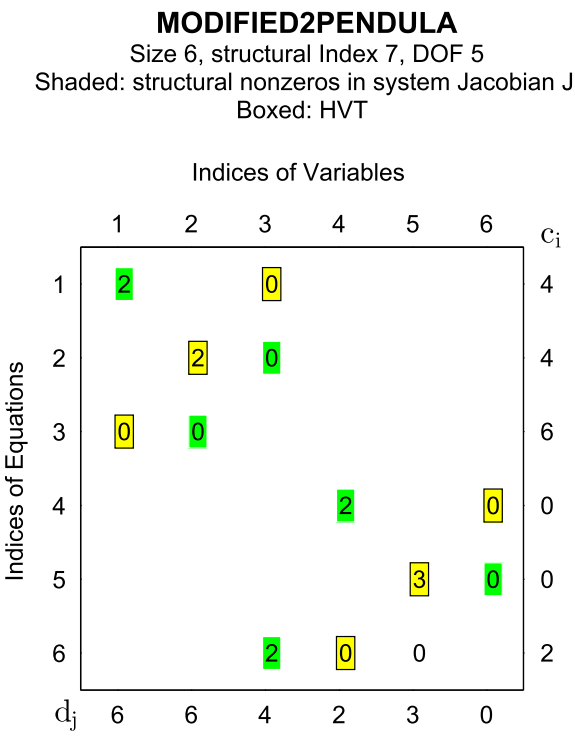


Fig. 1 Modified double pendulum structural analysis

DAE automatically. The solution of index 1 DAEs has been greatly studied in the literature and there are several tools that efficiently solve such problems, for example SUNDIALS [6], DASSL [15] and recently the MATLAB ODE solvers, such as ode15i, ode15s and ode25t.

3.2 The DD algorithm

Firstly we note that although originally the algorithm was given using a block lower triangular (BLT) form we shall, at least initially, consider the algorithm over one block. This simplifies notation and allows us to note interactions between blocks to find a reduced BTF with which to find DDs.

Consider a DAE to be written as $\mathcal{F}x = 0$, where $\mathcal{F}$ is a (column n-vector) differential-algebraic operator (DAO), following the notation of [7]:

1. $\nu(\mathcal{F})$, a column n-vector of non-negative integers, containing the minimum number of differentiations of each equation needed to get at least an index 1 DAE, usually found by Pantelides' Method [14] or equivalently the **c** in SA, see [18].
2. $D^{\nu} = \text{diag}(\frac{d^{\nu_1}}{dt^{\nu_1}}, \ldots, \frac{d^{\nu_n}}{dt^{\nu_n}})$, regarded as a DAO.
3. The differentiated problem $\mathcal{G}x = D^{\nu(\mathcal{F})}\mathcal{F}x = 0$.
4. A symbolic vector function, g, of equations in $\mathcal{G}x = 0$.

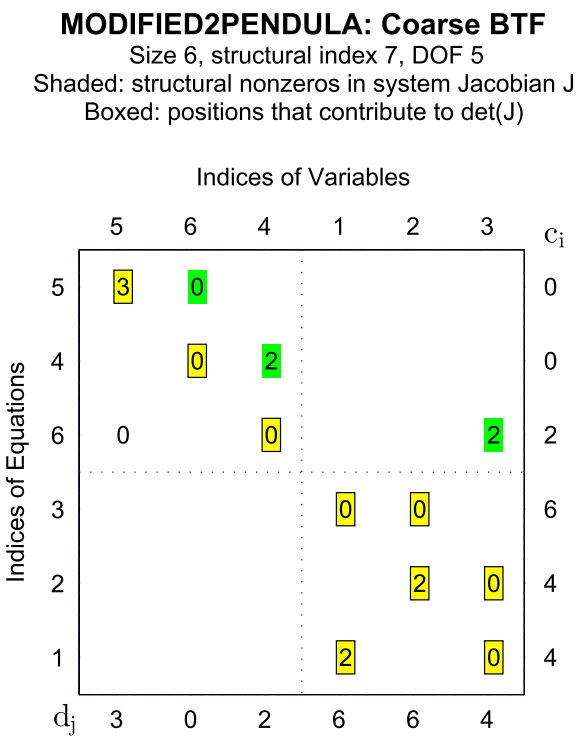


Fig. 2 Modified double pendulum structural analysis—coarse BTF

5. A symbolic vector, z, of highest order derivatives (HODs) of x in the differentiated problem, with first entry equal to the HOD of x_1 etc...
6. Equations in $\mathcal{G}x = 0$ are written as $g(z) = 0$

There are three main stages in finding DDs:

1. Get $\nu(\mathcal{F})$.
2. Obtain a differentiated problem $\mathcal{G}x = 0$.
3. Perform the index reduction algorithm. Loop through steps that select derivatives to be considered as algebraic variables in the solution process.

For ease of analysis later we will also assume, without loss of generality, that the equations (and variables) have been sorted into descending order with respect to number of differentiations needed, i.e. have been sorted with descending d_j (and c_i)—since $d_j \geq c_j$ this gives us each stage Jacobian a submatrix of previous stage Jacobians. We consider each stage by the superscript $[\kappa]$. To make it simpler to draw comparisons between SA and DDs we reorder the index reduction part of original algorithm as presented in [7] to Algorithm 1. The main changes between the original Algorithm and Algorithm 1 are a new numbering starting from 0 and removing the appearance of the matrix $M^{[\kappa]}$, since $G^{[\kappa+1]} = M^{[\kappa]}$. We also now compute H at the end of a stage as opposed to G, for more details see [11].

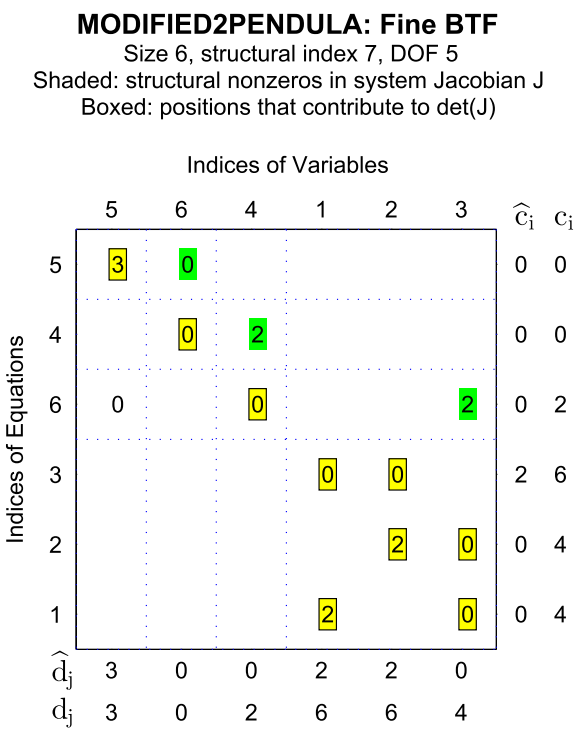


Fig. 3 Modified double pendulum structural analysis—fine BTF

Algorithm 1 The Reordered Dummy Derivative Algorithm

Require: $z = z^{[0]}$, $g(z) = g^{[0]}(z^{[0]})$, $\kappa = 0$
if $\kappa = 0$ **then**
  $G^{[0]} = \dfrac{\partial g^{[0]}}{\partial z^{[0]}}$
  Let m be the number of differentiated equations in $g^{[0]}(z^{[0]})$
  Let $H^{[0]}$ be the first m rows of $G^{[0]}$
  $\kappa = \kappa + 1$
else
  while $H^{[\kappa-1]} \neq [\ ]$ **do**
    Let $G^{[\kappa]}$ be m columns of $H^{[\kappa-1]}$ such that we have a non-singular matrix
    Make the corresponding variables used in $G^{[\kappa]}$ into DDs
    Omit one differentiation to get $z^{[\kappa]}$, $g^{[\kappa]}(z^{[\kappa]})$
    (where we only consider variables and equations in $G^{[\kappa]}$)
    Let m be the number of differentiated equations in $g^{[\kappa]}(z^{[\kappa]})$
    Let $H^{[\kappa]}$ be the first m rows of $G^{[\kappa]}$,...
    (the rows using differentiated equations in $g^{[\kappa]}(z^{[\kappa]})$)
    $\kappa = \kappa + 1$
Consider the new system using all equations $g^{[\kappa]}(z^{[\kappa]})$,where $\kappa \geq 0$,...
and dummy derivatives for $z^{[\kappa]}$, where $\kappa > 0$, as well as all original variables.

Example 5 We use an example found in [8] and apply Algorithm 1 to find DDs. Consider the linear constant coefficient DAE, with known smooth forcing functions $u_1(t), \ldots, u_4(t)$:

$$\mathcal{F}x = \mathcal{F}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \left.\begin{array}{llllll} f_1(t) = x_1 & +x_2 & & & +u_1(t) = 0 \\ f_2(t) = x_1 & +x_2 & +x_3 & & +u_2(t) = 0 \\ f_3(t) = x_1 & & +\dot{x}_3 & +x_4 & +u_3(t) = 0 \\ f_4(t) = 2\ddot{x}_1 & +\ddot{x}_2 & +\ddot{x}_3 & +\ddot{x}_4 & +u_4(t) = 0 \end{array}\right\}. \tag{9}$$

We need a number of differentiations for each equation, to do this we'll compute the signature matrix and find $\mathbf{c}$:

$$\Sigma = \begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ d_j \end{array} \begin{array}{c} \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \end{array} \\ \begin{pmatrix} 0^\bullet & 0 & -\infty & -\infty \\ 0 & 0^\bullet & 0 & -\infty \\ 0 & -\infty & 1^\bullet & 0 \\ 2 & 2 & 2 & 1^\bullet \end{pmatrix} \\ \begin{array}{cccc} 2 & 2 & 2 & 1 \end{array} \end{array} \begin{array}{c} c_i \\ 2 \\ 2 \\ 1 \\ 0 \end{array}.$$

The HODs are $z^{[0]} = (\ddot{x}_1, \ddot{x}_2, \ddot{x}_3, \dot{x}_4)$, the current equations are $g^{[0]}(z^{[0]}) = (\ddot{f}_1, \ddot{f}_2, \dot{f}_3, f_4)^T$.

Stage 0: Initialise and then remove undifferentiated equations, since $m = 3$ we get:

$$G^{[0]} = \frac{\partial g^{[0]}}{\partial z^{[0]}} = \begin{array}{c} \\ \ddot{f}_1 \\ \ddot{f}_2 \\ \dot{f}_3 \\ d \end{array} \begin{array}{c} \begin{array}{cccc} \ddot{x}_1 & \ddot{x}_2 & \ddot{x}_3 & \dot{x}_4 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 2 & 1 & 1 & 1 \end{pmatrix} \end{array} \quad \text{and}$$

$$H^{[0]} = \begin{array}{c} \\ \ddot{f}_1 \\ \ddot{f}_2 \\ \dot{f}_3 \end{array} \begin{array}{c} \begin{array}{cccc} \ddot{x}_1 & \ddot{x}_2 & \ddot{x}_3 & \dot{x}_4 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \end{array}.$$

Stage 1: We now have two possibilities for selecting columns to get a non-singular matrix. Choosing to omit column 2 gives $G^{[1]}$ below. Therefore $\ddot{x}_1$, $\ddot{x}_3$ and $\dot{x}_4$ are made DDs, denoted by x_1'', x_3'' and x_4'. Reducing the order of differentiation by 1 gives $z^{[1]} = (\dot{x}_1, \dot{x}_3, x_4)$, current equations are $g^{[1]}(z^{[1]}) = (\dot{f}_1, \dot{f}_2, f_3)^T$, so $m = 2$ and we get $H^{[1]}$ as below:

$$G^{[1]} = \begin{array}{c} \\ \ddot{f}_1 \\ \ddot{f}_2 \\ \dot{f}_3 \end{array} \begin{array}{c} \begin{array}{ccc} \ddot{x}_1 & \ddot{x}_3 & \dot{x}_4 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \end{array} \quad \text{and} \quad H^{[1]} = \begin{array}{c} \\ \dot{f}_1 \\ \dot{f}_2 \end{array} \begin{array}{c} \begin{array}{ccc} \dot{x}_1 & \dot{x}_3 & x_4 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \end{array}.$$

Stage 2: We now have only one possibility, so must have:

$$G^{[2]} = \begin{array}{c} \\ \dot{f}_1 \\ \dot{f}_2 \end{array} \begin{array}{c} \begin{array}{cc} \dot{x}_1 & \dot{x}_3 \end{array} \\ \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \end{array}.$$

Thus $\dot{x}_1$ and $\dot{x}_3$ are made DDs, denoted by x_1' and x_3'. Reducing the order of differentiation by 1 gives $z^{[2]} = (x_1, x_3)$, the current equations are $g^{[2]}(z^{[2]}) = (f_1, f_2)^T$, so $m = 0$ and $H^{[2]} = [\ ]$ and the algorithm ends, yielding an index 1 system equivalent to the one found in [7]:

$$\left.\begin{array}{llll} f_1(t) = x_1 & +x_2 & + u_1(t) & = 0 \\ f_2(t) = x_1 & +x_2 + x_3 & + u_2(t) & = 0 \end{array}\right\}$$
$$\left.\begin{array}{llll} \dot{f}_1(t) = x_1' & +\dot{x}_2 & + \dot{u}_1(t) & = 0 \\ \dot{f}_2(t) = x_1' & +\dot{x}_2 + x_3' & + \dot{u}_2(t) & = 0 \\ f_3(t) = x_1 & + x_3' & +x_4 + u_3(t) & = 0 \end{array}\right\}$$
$$\left.\begin{array}{llll} \ddot{f}_1(t) = x_1'' & +\ddot{x}_2 & + \ddot{u}_1(t) & = 0 \\ \ddot{f}_2(t) = x_1'' & +\ddot{x}_2 + x_3'' & + \ddot{u}_2(t) & = 0 \\ \dot{f}_3(t) = x_1' & + x_3'' & +x_4' + \dot{u}_3(t) & = 0 \\ f_4(t) = 2x_1'' & +\ddot{x}_2 + x_3'' & +x_4' + u_4(t) & = 0 \end{array}\right\} \tag{10}$$

The zeros in the third column of $H^{[1]}$ mean we never choose x_4 as a DD, this is always the case for undifferentiated variables, as explained in the following section. Also, the variables and equations used in $G^{[\kappa]}$ are just $Dz^{[\kappa]}$ and $Dg^{[\kappa]}(z^{[\kappa]})$.

4 Structural analysis based dummy derivatives

If we consider $H^{[\kappa]}$ to be a matrix of size $n_\kappa \times m_\kappa$, then from the algorithm in Sect. 3 we have $\binom{m_\kappa}{n_\kappa}$ potential index 1 systems at stage $\kappa + 1$. Thus the potential number of index 1 systems obtainable by the method can be very large for practical examples. We would like to use the structural analysis of Sect. 2 to inform our choice of DDs and thus limit the potential number of systems considered at each stage. See Sect. 5 for other ways of achieving numerical speed up.

4.1 Similarities in the methods

There are several similarities between Signature Matrix based SA and DDs. Firstly, as was said above, the $\nu(\mathcal{F})$ used in DDs is the same as $\mathbf{c}$ in SA, as from [18] we see that Pantelides algorithm [14] and SA can be used interchangeably. Therefore we have that $D^{\nu(\mathcal{F})} = \mathrm{diag}(\frac{d^{c_1}}{dt^{c_1}}, \ldots, \frac{d^{c_n}}{dt^{c_n}})$ and the following equality:

$$\mathcal{G}x = D^{\nu(\mathcal{F})}\mathcal{F}x = D^{\mathbf{c}}\mathcal{F}x.$$

We are differentiating each equation f_i, c_i times, so the maximum derivative for each variable x_j in $\mathcal{G}x = 0$ will equal $\max_i(\sigma_{i,j} + c_i)$; from (1) this is d_j. Hence, the 0^{th} stage system in DDs is the 0^{th} stage system in SA. Thus the differentiated problem can be written:

$$f_i^{(c_i)}(t, x_j^{(d_j)};\ \text{lower order derivatives}\ ) = 0, \quad \text{for } i, j = 1, \ldots, n.$$

Therefore we must have that:

$$z^{[0]} = (x_1^{(d_1)}, \ldots, x_n^{(d_n)}).$$

The formula for the DD Jacobian matrix $G^{[0]}$ can now be written in this SA based notation to show it actually equals the stage 0 SA Jacobian $\mathbf{J}$.

$$G^{[0]} = \frac{\partial g^{[0]}}{\partial z^{[0]}} = \left(\frac{\partial f_i^{(c_i)}}{\partial x_j^{(d_j)}}\right) = \mathbf{J}. \tag{11}$$

Going to the next stage in DDs by reducing the order of differentiation by 1 is equivalent to reducing the offset vector $\mathbf{c}$ by 1 (and consequently reducing $\mathbf{d}$ by 1 also). Therefore at stage 1 in DDs we will be considering the equations used in stage -1 of SA, since SA increases the order of differentiation by one at each stage. This leads us to the following observation.

Lemma 2 *The equations used at stage k in SA are equal to those used at stage $\kappa = -k$ in DDs (when writing down $G^{[\kappa]}$), for each stage k between k_{min} and 0.*

Proof We have already shown that at stage 0 both methods use the same equations. In DDs we now remove all equations such that $c_i = 0$. We then omit one differentiation and repeat. Hence we remove, at stage 1, equations such that $c_i - 1 = 0$, and by induction, at stage κ equations such that $c_i - \kappa = 0$, where κ is the stage number. From (3) these are exactly the equations considered at stage $-\kappa = k$ in SA. □

Due to the above theorem we will now use the term 'equivalent stage' to mean DD stage κ when talking about SA stage $-k = \kappa$ and vice versa. We now take notation from [18] in order to write down the k^{th} stage Structural Jacobian found in SA. Consider each variable x_j as a function of an independent variable t and let x_{jl} represent $x_j^{(l)}(t)$. Then for an $n \times n$ DAE we have the index set:

$$\mathcal{J} = \{(j, l) \mid j = 1, \ldots, n; l = 0, 1, \ldots\},$$

similarly for the equations we use the set:

$$\mathcal{I} = \{(i, l) \mid i = 1, \ldots, n; l = 0, 1, \ldots\}.$$

Table 2 The results of SA on the linear DAE (9)—also showing DDs by a prime

k	Equations being used	Variables being found
-2	f_1, f_2	x_1, x_2, x_3
-1	$\dot{f}_1, \dot{f}_2, f_3$	$x_1', \dot{x}_2, x_3', x_4$
0	$\ddot{f}_1, \ddot{f}_2, \dot{f}_3, f_4$	$x_1'', \ddot{x}_2, x_3'', x_4'$

This gives us a notation for the variables used at each stage in the SA, that is:

$$I_k = \{(i,l) \in \mathcal{I} \mid l = k + c_i\}$$
$$J_k = \{(j,l) \in \mathcal{J} \mid l = k + d_j\}$$

where the offsets are taken to be canonical unless otherwise stated. At each SA stage we have:

$$m_k = \mid \mathcal{I}_k \mid = \mid \{j \mid d_j + k \geq 0\} \mid, \quad n_k = \mid \mathcal{J}_k \mid = \mid \{i \mid c_i + k \geq 0\} \mid .$$

We write f_{I_k} to mean the set of equations used at stage k in SA and $f_{I_{\leq k}}$ to mean the set of equations used between k stage $k_{\min}$ and k—i.e. $f_{I_{k_{\min}}} \cup f_{I_{k_{\min}+1}} \cup \cdots \cup f_{I_k}$, similarly for the variables.

Again, from [18] we have that the system Jacobian used at stage k in SA is given by:

$$\mathbf{J}_k = \frac{\partial f_{I_k}}{\partial x_{J_k}}. \tag{12}$$

Recall the note at the end of Sect. 3.2, giving us the following Lemma:

Lemma 3 *In DDs if at stage κ, $d_j = \kappa + 1$ then column j cannot be in $G^{[\kappa+1]}$.*

Proof If $1 \leq i \leq m_k$ and $j > m_k$ then $k + c_i > 0$ and $k + d_j \leq 0$, hence $d_j - c_i < 0$ and thus cannot be equal to $\sigma_{i,j}$, so that $(\mathbf{J}_k)_{ij} = 0$ due to the definition of the System Jacobian in Equation (5). Since we have $G^{[0]} = \mathbf{J}_0 = \mathbf{J}$ and the DD algorithm is reducing the order of differentiation by one at each stage, if the column referring to x_j appears in $\mathbf{J}_k$ and $H^{[-k-1]}$ then its entries must be the same. Thus columns with $d_j = \kappa + 1$ cannot be selected to form $G^{[-k]}$, as they will be columns of structural zeros. □

Thus columns representing variables that are undifferentiated cannot be chosen as DDs, as one would expect.

Example 6 Consider example 5 and recall the index 1 system given in Eq. (10). Compare this with the SA results in Table 2, the variables that became DDs are marked by prime notation. In Example 5 we make a subset of the variables found at stage k in SA into DDs at stage $-k + 1 = \kappa + 1$ in the DD scheme, using the same equations in both cases, see Table 2.

The total number of DDs introduced will be $\sum_i c_i$, since this is the total number of new equations introduced and we identify one DD with each new equation. At each stage the variables that produced DDs are a subset of the variables that produced DDs at the previous stage (necessarily excluding those with $d_j = \kappa + 1$), of size m_{k-1}, with each variable being differentiated one time less than in the previous stage. So, the DDs will be a subset of the variables solved for at the equivalent stage $+1$ of SA, thus we have the following:

Theorem 2 *The matrix $G^{[\kappa]}$ is a submatrix of (it may be equal to) $\mathbf{J}_k$, where $\kappa = -k$.*

Before going in to a deeper comparison we consider the 0 DOF case.

4.2 DDs and SA in the 0 DOF case

We begin with the following observation.

Lemma 4 *If a square non-singular DAE has 0 DOF and a HVT on the main diagonal of Σ then $\mathbf{d} = \mathbf{c}^T$.*

Proof Immediate from (1) equality on a HVT constraint. □

This gives us the following Theorem:

Theorem 3 *If we have 0 DOF then $G^{[-k]} = \mathbf{J}_k$ for each stage k between k_{min} and 0.*

Proof We have no choice in our selection of $G^{[\kappa]}$ since $H^{[\kappa-1]}$ (of size $m \times n$ say) must contain only m columns of structural non-zeros, since $n-m$ other columns correspond to undifferentiated variables due to Lemma 4. Noting that $\mathbf{J}_0 = G^{[0]}$ completes the proof. □

Example 7 In [2] the authors introduce a DAE for modelling a robot arm. It is reformulated to be SWP (i.e. have finite Val(Σ)) in [17]. We shall use the 6×6 formulation introduced in [17] and arrange the equations and variables such that we clearly illustrate the block triangular structure of the problem. Structural Jacobian, signature matrix and offsets (with a HVT marked by $\bullet$ and $-\infty$ entries left blank) for this DAE are, where F_w means $\partial F/\partial w$ and so on:

$$\Sigma = \begin{array}{c} \\ G \\ H \\ D \\ F \\ E \\ K \\ d_j \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_3 & w & x_2 & u_2 & u_1 \end{array} \\ \begin{pmatrix} 0^\bullet & 0 & & & & \\ 0 & 0^\bullet & & & & \\ 2 & 1 & 0^\bullet & 0 & & \\ 1 & 2 & 0 & 0^\bullet & & \\ 1 & 1 & 0 & 2 & 0^\bullet & \\ & & 0 & & 0 & 0^\bullet \end{pmatrix} \\ \begin{array}{cccccc} 4 & 4 & 2 & 2 & 0 & 0 \end{array} \end{array} \begin{array}{c} c_i \\ 4 \\ 4 \\ 2 \\ 2 \\ 0 \\ 0 \end{array},$$

$$\mathbf{J} = \begin{array}{c} \\ G^{(4)} \\ H^{(4)} \\ \ddot{D} \\ \ddot{F} \\ E \\ K \end{array} \begin{array}{c} \begin{array}{cccccc} x_1^{(4)} & x_3^{(4)} & \ddot{w} & \ddot{x}_2 & u_2 & u_1 \end{array} \\ \begin{pmatrix} G_{x_1} & G_{x_3} & 0 & 0 & 0 & 0 \\ H_{x_1} & H_{x_3} & 0 & 0 & 0 & 0 \\ D_{\ddot{x}_1} & 0 & D_w & D_{x_2} & 0 & 0 \\ 0 & F_{\ddot{x}_3} & F_w & F_{x_2} & 0 & 0 \\ 0 & 0 & 0 & E_{\ddot{x}_2} & E_{u_2} & 0 \\ 0 & 0 & 0 & 0 & K_{u_2} & K_{u_1} \end{pmatrix} \end{array}.$$

For later observations it's useful to note the above signature matrix has four coarse (also fine) blocks, two of size 2×2 and two of size 1×1. Working through the DD algorithm yields $\nu(\mathcal{F}) = (4, 4, 2, 2, 0, 0)$, and thus the differentiated system $\mathcal{G}$ is:

$$\begin{aligned} G^{(4)} &= 0, & H^{(4)} &= 0, \\ D^{(2)} &= 0, & F^{(2)} &= 0, \\ E &= 0, & K &= 0. \end{aligned}$$

Stage 0: The vector of HODs is $z^{[0]} = (x_1^{(4)}, x_3^{(4)}, \ddot{w}, \ddot{x}_2, u_2, u_1)^T$ and $g^{[0]} = (G^{(4)}, H^{(4)}, D^{(2)}, F^{(2)}, E, K)^T$.

Thus we have a DD Jacobian of the form:

$$\frac{\partial g^{[0]}}{\partial z^{[0]}} = G^{[0]} = \begin{array}{c} \\ G^{(4)} \\ H^{(4)} \\ \ddot{D} \\ \ddot{F} \\ E \\ K \\ d_j \end{array} \begin{array}{c} \begin{array}{cccccc} x_1^{(4)} & x_3^{(4)} & \ddot{w} & \ddot{x}_2 & u_2 & u_1 \end{array} \\ \begin{pmatrix} G^{(4)}_{x_1^{(4)}} & G^{(4)}_{x_3^{(4)}} & 0 & 0 & 0 & 0 \\ H^{(4)}_{x_1^{(4)}} & H^{(4)}_{x_3^{(4)}} & 0 & 0 & 0 & 0 \\ \ddot{D}_{x_1^{(4)}} & 0 & \ddot{D}_{\ddot{w}} & \ddot{D}_{\ddot{x}_2} & 0 & 0 \\ 0 & \ddot{F}_{x_3^{(4)}} & \ddot{F}_{\ddot{w}} & \ddot{F}_{\ddot{x}_2} & 0 & 0 \\ 0 & 0 & 0 & E_{\ddot{x}_2} & E_{u_2} & 0 \\ 0 & 0 & 0 & 0 & K_{u_2} & K_{u_1} \\ 4 & 4 & 2 & 2 & 0 & 0 \end{pmatrix} \end{array} \begin{array}{c} c_i \\ 4 \\ 4 \\ 2 \\ 2 \\ 0 \\ 0 \end{array}.$$

By Griewank's Lemma 1 this is equivalent to $\mathbf{J}$. Removing equations with $c_i = 0$ yields:

$$H^{[0]} = \begin{array}{c} \\ G^{(4)} \\ H^{(4)} \\ \ddot{D} \\ \ddot{F} \\ d_j \end{array} \begin{array}{c} \begin{array}{cccccc} x_1^{(4)} & x_3^{(4)} & \ddot{w} & \ddot{x}_2 & u_2 & u_1 \end{array} \\ \begin{pmatrix} G^{(4)}_{x_1^{(4)}} & G^{(4)}_{x_3^{(4)}} & 0 & 0 & 0 & 0 \\ H^{(4)}_{x_1^{(4)}} & H^{(4)}_{x_3^{(4)}} & 0 & 0 & 0 & 0 \\ \ddot{D}_{x_1^{(4)}} & 0 & \ddot{D}_{\ddot{w}} & \ddot{D}_{\ddot{x}_2} & 0 & 0 \\ 0 & \ddot{F}_{x_3^{(4)}} & \ddot{F}_{\ddot{w}} & \ddot{F}_{\ddot{x}_2} & 0 & 0 \\ 4 & 4 & 2 & 2 & 0 & 0 \end{pmatrix} \end{array} \begin{array}{c} c_i \\ 4 \\ 4 \\ 2 \\ 2 \end{array}.$$

Stage 1: We are now forced to remove the last two columns of $H^{[0]}$ to get a non-singular matrix, choosing $x_1^{(4)},\ x_3^{(4)},\ \ddot{w},\ \ddot{x}_2$ as DDs and reducing the order of differentiation:

$$G^{[1]} = \begin{array}{c} \\ G^{(4)} \\ H^{(4)} \\ \ddot{D} \\ \ddot{F} \\ d_j \end{array} \begin{array}{c} \begin{array}{cccc} x_1^{(4)} & x_3^{(4)} & \ddot{w} & \ddot{x}_2 \end{array} \\ \begin{pmatrix} G^{(4)}_{x_1^{(4)}} & G^{(4)}_{x_3^{(4)}} & 0 & 0 \\ H^{(4)}_{x_1^{(4)}} & H^{(4)}_{x_3^{(4)}} & 0 & 0 \\ \ddot{D}_{x_1^{(4)}} & 0 & \ddot{D}_{\ddot{w}} & \ddot{D}_{\ddot{x}_2} \\ 0 & \ddot{F}_{x_3^{(4)}} & \ddot{F}_{\ddot{w}} & \ddot{F}_{\ddot{x}_2} \end{pmatrix} \\ \begin{array}{cccc} 4 & 4 & 2 & 2 \end{array} \end{array} \begin{array}{c} c_i \\ 4 \\ 4 \\ 2 \\ 2 \end{array},$$

$$H^{[1]} = \begin{array}{c} \\ G^{(3)} \\ H^{(3)} \\ \dot{D} \\ \dot{F} \\ d_j \end{array} \begin{array}{c} \begin{array}{cccc} x_1^{(3)} & x_3^{(3)} & \dot{w} & \dot{x}_2 \end{array} \\ \begin{pmatrix} G^{(3)}_{x_1^{(3)}} & G^{(3)}_{x_3^{(3)}} & 0 & 0 \\ H^{(3)}_{x_1^{(3)}} & H^{(3)}_{x_3^{(3)}} & 0 & 0 \\ \dot{D}_{x_1^{(3)}} & 0 & \dot{D}_{\dot{w}} & \dot{D}_{\dot{x}_2} \\ 0 & \dot{F}_{x_3^{(3)}} & \dot{F}_{\dot{w}} & \dot{F}_{\dot{x}_2} \end{pmatrix} \\ \begin{array}{cccc} 3 & 3 & 1 & 1 \end{array} \end{array} \begin{array}{c} c_i \\ 3 \\ 3 \\ 1 \\ 1 \end{array}.$$

Stage 2: Since $H^{[1]}$ is square $G^{[2]} = H^{[1]}$. As $c_i - \kappa = 0$ no rows (and therefore no columns) are removed at this stage, so that:

$$H^{[2]} = \begin{array}{c} \\ \ddot{G} \\ \ddot{H} \\ D \\ F \\ d_j \end{array} \begin{array}{c} \begin{array}{cccc} \ddot{x}_1 & \ddot{x}_3 & w & x_2 \end{array} \\ \begin{pmatrix} \ddot{G}_{\ddot{x}_1} & \ddot{G}_{\ddot{x}_3} & 0 & 0 \\ \ddot{H}_{\ddot{x}_1} & \ddot{H}_{\ddot{x}_3} & 0 & 0 \\ D_{\ddot{x}_1} & 0 & D_w & D_{x_2} \\ 0 & F_{\ddot{x}_3} & F_w & F_{x_2} \end{pmatrix} \\ \begin{array}{cccc} 2 & 2 & 0 & 0 \end{array} \end{array} \begin{array}{c} c_i \\ 2 \\ 2 \\ 0 \\ 0 \end{array}.$$

Stage 3: Again, $H^{[2]}$ is already square so $G^{[3]} = H^{[2]}$.

$$H^{[3]} = \begin{array}{c} \\ \dot{G} \\ \dot{H} \\ d_j \end{array} \begin{array}{c} \begin{array}{cccc} \dot{x}_1 & \dot{x}_3 & w & x_2 \end{array} \\ \begin{pmatrix} \dot{G}_{\dot{x}_1} & \dot{G}_{\dot{x}_3} & 0 & 0 \\ \dot{H}_{\dot{x}_1} & \dot{H}_{\dot{x}_3} & 0 & 0 \end{pmatrix} \\ \begin{array}{cccc} 1 & 1 & 0 & 0 \end{array} \end{array} \begin{array}{c} c_i \\ 1 \\ 1 \end{array},$$

Table 3 DDs and SA stages for the robot arm

DD stage	SA stage	Equations being used	Variables being found	DDs selected
4	−4	G, H	x_1, x_3	x_1', x_3'
3	−3	$\dot{G}, \dot{H}$	$\dot{x}_1, \dot{x}_3$	x_1'', x_3''
2	−2	$\ddot{G}, \ddot{H}, D, F$	$\ddot{x}_1, \ddot{x}_3, w, x_2$	$x_1^{(3)}, x_3^{(3)}, w', x_2'$
1	−1	$G^{(3)}, H^{(3)}, \dot{D}, \dot{F}$	$x_1^{(3)}, x_3^{(3)}, \dot{w}, \dot{x}_2$	$x_1^{(4)}, x_3^{(4)}, w'', x_2''$
0	0	$G^{(4)}, H^{(4)}, \ddot{D}, \ddot{F}, E, K$	$x_1^{(4)}, x_3^{(4)}, \ddot{w}, \ddot{x}_2, u_2, u_1$	N/A

Stage 4: Then:

$$G^{[4]} = \begin{array}{c} \\ \dot{G} \\ \dot{H} \\ d_j \end{array} \begin{array}{c} \begin{array}{cc} \dot{x}_1 & \dot{x}_3 \end{array} \\ \begin{pmatrix} \dot{G}_{\dot{x}_1} & \dot{G}_{\dot{x}_3} \\ \dot{H}_{\dot{x}_1} & \dot{H}_{\dot{x}_3} \end{pmatrix} \\ \begin{array}{cc} 1 & 1 \end{array} \end{array} \begin{array}{c} c_i \\ 1 \\ 1 \end{array}$$

Finally we get $H^{[4]} = [\ ]$ and the algorithm terminates. By Griewank's Lemma 1 we have $G^{[-k]} = \mathbf{J}_k$ for all k between −4 and 0 inclusively. We list a comparison between the SA and DD algorithms in Table 3.

The DDs are equivalent to the differentiated variables solved for at each prior stage in SA as expected, due to the 0 DOF in this example.

4.3 Structurally necessary dummy derivatives

Because the equations used in each equivalent stage of DDs and SA are the same and variables in each DD stage are a subset of those in the SA stage we have the following Theorem:

Theorem 4 *If we have an equal number of variables n_k and equations m_k used at stage k in the SA then we will have no choice when finding $G^{[-k+1]}$ in the DD scheme, i.e. $H^{[\kappa]}$ is square.*

Corollary 1 *If m_k=n_k in the SA scheme at stage k then all subsequent derivatives of variables in $z^{[k]}$ used by the DD scheme at stage $\kappa = -k$ must be DDs in the final index 1 system, i.e. $Dz^{[k]}, \ldots, D^{(-k)}z^{[k]}$ must be DDs.*

Definition 2 *(Structurally necessary DDs)* If there exists a k such that $m_k = n_k$ and $d_j - k \geq 0$ then $x_j^{(d_j-k+1)}, \ldots, x_j^{(d_j)}$ are termed structurally necessary dummy derivatives.

This gives us the following improved DD algorithm, where we can identify structurally necessary DDs without computing numerical Jacobians:

Algorithm 2 $m_K = n_K$ Algorithm

1: **for** $K = k_{\min} : -1$ **do**
2: Find SA solution scheme
3: Note stages where $m_K = n_K$
4: Make subsequent derivatives of such variables DDs
5: **for** $K = 0 : -k_{\min}$ **do**
6: Work through DDs algorithm, but:
7: keep columns for already known DDs from step 4 when finding $G^{[K]}$

For 0 DOF systems this identifies all DDs as one might expect. Clearly Algorithm 2 finds all structurally necessary DDs. Applying the first half of Algorithm 2 (finding structurally necessary DDs) to the robot arm gives us the following staged solution scheme, produced by an extension (not yet released) to the authors' code DAESA [10]:

```
 Do a pass through the Structural Analysis scheme:
K = -4: Make the following derivatives into dummy
        derivatives
        x1', x1'', x1''', x1'''', x2', x2'', x2''',
        x2''''
K = -3: No dummy derivatives can be discovered
        structurally at this stage
K = -2: Make the following derivatives into dummy
        derivatives
        x3', x3'', x4', x4''
K = -1: No dummy derivatives can be discovered
        structurally at this stage
```

One should compare this with Table 3 to convince themselves of the result.

Example 8 Recall the modified double pendulum DAE (8) and the associated signature matrix and offsets in Fig. 1. Table 4 gives the SA solution stages for this problem. In Table 4 we see all higher order derivatives of variables found at stage 4 in DDs will be made DDs (i.e. those used in SA stages $-3, -2, -1$ or DD stages 1, 2, 3). Consider now the different BTFs as shown in Fig. 1. Applying Corollary 1 corresponds to us

Table 4 SA stages for Eq. (8)

SA stage	m_k	n_k
-6	1	2
-5	1	2
-4	3	3
-3	3	4
-2	4	5
-1	4	5
0	6	6

Table 5 Dummy derivative stages for Eq. (8)

Dummy derivative stage κ	$z^{[\kappa]}$	$g^{[\kappa]}(z^{[\kappa]})$
0	$(x_1^{(6)}, x_2^{(6)}, x_3^{(4)}, x_5^{(3)}, \ddot{x}_4, x_6)$	$(f_3^{(6)}, f_1^{(4)}, f_2^{(4)}, \ddot{f}_6, f_4, f_5)^T$
1	$(x_1^{(5)}, x_2^{(5)}, x_3^{(3)}, \ddot{x}_5)$	$(f_3^{(5)}, f_1^{(3)}, f_2^{(3)}, \dot{f}_6)^T$
2	$(x_1^{(4)}, x_2^{(4)}, \ddot{x}_3, \dot{x}_5)$	$(f_3^{(4)}, \ddot{f}_1, \ddot{f}_2, f_6)^T$
3	$(x_1^{(3)}, x_2^{(3)}, \dot{x}_3, x_5)$	$(f_3^{(3)}, \dot{f}_1, \dot{f}_2)^T$
4	$(\ddot{x}_1, \ddot{x}_2, x_3)$	$(\ddot{f}_3, f_1, f_2)^T$
5	$(\dot{x}_1)$	$(\dot{f}_3)^T$
6	(x_1)	$(f_3)^T$

solving the first coarse block as a stand alone system and then using it to solve the second coarse block. Compare this with the $z^{[\kappa]}$ and $g^{[\kappa]}(z^{[\kappa]})$ found in DDs in Table 5 (we have re-ordered equations and variables so they correspond with the coarse block ordering).

Due to our ordering in the DD algorithm we introduce DDs for $Dz^{[\kappa]}$ at stage κ, e.g. at stage 4 we are left with the 3×3 system given by the first coarse block in our BTF as expected. We note this algorithm for structurally necessary DDs looks similar to solving for DDs based on the coarse BTF, see Sect. 4.4 for why this is not quite true.

Example 9 This improved DD algorithm does indeed achieve our goal of reducing the total number of potentially needed index 1 systems: Consider again the DAE (8), working through the structural analysis we see that at stage $k = -4$ we have $m_k = n_k$. At stage -4 we are solving for $\ddot{x}_1, \ddot{x}_2, x_3$, so we know to keep columns corresponding to these variables when working through DDs. For example, with $\bullet$ indicating a structural non-zero and a blank indicating a structural zero we have $G^{[0]}$ and $H^{[0]}$:

$$
\begin{array}{c}
 \\
f_1^{(4)} \\
f_2^{(4)} \\
f_3^{(6)} \\
f_4 \\
f_5 \\
\ddot{f}_6 \\
d_j
\end{array}
\begin{array}{c}
\begin{array}{cccccc} x_1^{(6)} & x_2^{(6)} & x_3^{(4)} & \ddot{x}_4 & x_5^{(3)} & x_6 \end{array} \\
\left(\begin{array}{cccccc}
\bullet & & \bullet & & & \\
 & \bullet & \bullet & & & \\
\bullet & \bullet & & & & \\
 & & & \bullet & & \bullet \\
 & & & & \bullet & \bullet \\
 & & \bullet & \bullet & & \\
6 & 6 & 4 & 2 & 3 & 0
\end{array}\right)
\end{array}
\begin{array}{c}
c_i \\
4 \\
4 \\
6 \\
0 \\
0 \\
2
\end{array}
$$

$$
\begin{array}{c}
\\
f_1^{(4)} \\
f_2^{(4)} \\
f_3^{(6)} \\
f_6 \\
d_j
\end{array}
\begin{array}{c}
\begin{array}{cccccc} x_1^{(6)} & x_2^{(6)} & x_3^{(4)} & \ddot{x}_4 & x_5^{(3)} & x_6 \end{array} \\
\begin{pmatrix}
\bullet & & \bullet & & & \\
 & \bullet & \bullet & & & \\
\bullet & \bullet & & & & \\
 & & \bullet & \bullet & & \\
6 & 6 & 4 & 2 & 3 & 0
\end{pmatrix}
\end{array}
\begin{array}{c} c_i \\ 4 \\ 4 \\ 6 \\ 2 \end{array} .
$$

We must keep the first three columns and hence only have to check 3 matrices for non-singularity (in practice best condition number), as opposed to the 15 we would otherwise have to check—although clearly in this case inspection tells us we choose the first 4 columns.

4.4 Block based dummy derivatives

Using a block decomposition may yield a way of reducing the size of potential $G^{[\kappa]}$ matrices at each stage, which should offer computational speed up when checking the condition number of each Jacobian when doing dummy pivoting, see Sect. 5. Before giving an algorithm for finding DDs on blocks we ask if Algorithm 2 was already doing something similar to a BTF for us. We consider a fine block decomposition (which is itself a BTF within the coarse BTF) and ask if we could further reduce potential index 1 choices when considering m_k being equal to n_k during our SA stages.

Theorem 5 *If given an $n \times n$ DAE and there exists a $k \in \{k_{min}, \ldots, -1\}$ such that $n_k = m_k = \mu$ for some $0 < \mu < n$, then the DAE must decompose in to at least 2 coarse blocks of size μ and $(n - \mu)$.*

Proof Similar to the proof of Lemma 3 one partitions the matrix into the following and then notes the top right block is empty by (1).

$$
\Sigma = \left[\begin{array}{c|c}
\begin{array}{c} k + d_j \geq 0 \\ k + c_i \geq 0 \end{array} & \begin{array}{c} k + d_j < 0 \\ k + c_i \geq 0 \end{array} \\
\hline
\begin{array}{c} k + d_j \geq 0 \\ k + c_j < 0 \end{array} & \begin{array}{c} k + d_j < 0 \\ k + c_j < 0 \end{array}
\end{array}\right]
\begin{array}{c} 1 \\ \vdots \\ \mu \\ \mu + 1 \\ \vdots \\ n \end{array} .
$$

$$
\begin{array}{cccccc} 1 & \ldots & \mu & \mu + 1 & \ldots & n \end{array}
$$

□

This means, we cannot have a coarse block irreducible DAE with $n_k = m_k$, unless $k \geq 0$ or $k < k_{\min}$ and therefore $n_k = m_k$ never occurs when solving via fine blocks

unless the DAE has m fine blocks and we are solving fine block m, i.e. are at global stage 0. Hence we turn our attention to the interactions between blocks rather than trying to optimise further within a block. We define local offsets associated with an arbitrary block form:

Definition 3 (*Block Local Offsets*) We denote the canonical local offsets associated with an arbitrary block form, were its blocks treated as a stand alone systems, as $\check{c}_i$ and $\check{d}_j$.

Given a block form of Σ with L blocks we have Algorithm 3. In the following

Algorithm 3 Block based Dummy Derivatives

1: **for** $l = 1, \ldots, L$ **do**
Require: $z_l^{[0]}, g_l^{[0]}(z_l^{[0]}), \kappa_l = 0$
2: **if** $j = 0$ **then**
3: $G_l^{[0]} = \dfrac{\partial g_l^{[0]}}{\partial z_l^{[0]}}$
4: Let m be the number of differentiated equations in $g_l^{[0]}(z_l^{[0]})$
5: Let $H_l^{[0]}$ be the first m rows of $G_l^{[0]}$
6: $\kappa = \kappa + 1$
7: **else**
8: **while** $H_l^{[\kappa_l - 1]} \neq [\ ]$ **do**
9: Let $G_l^{[\kappa_l]}$ be m columns of $H_l^{[\kappa_l - 1]}$ such that we have a non-singular matrix
10: Make the corresponding variables used in $G_l^{[\kappa_l]}$ DDs
11: Omit one differentiation to get $z_l^{[\kappa_l]}, g_l^{[\kappa_l]}(z_l^{[\kappa_l]})$,...
12: (where we only consider variables and equations in $G_l^{[\kappa_l]}$)
13: Let m be the number of differentiated equations in $g_l^{[\kappa_l]}(z_l^{[\kappa_l]})$
14: Let $H_l^{[\kappa_l]}$ be the first m rows of $G_l^{[\kappa_l]}$,...
15: (the rows using differentiated equations in $g_l^{[\kappa_l]}(z_l^{[\kappa_l]})$)
16: $\kappa_l = \kappa_l + 1$
17: Consider the new system using all equations $g_l^{[\kappa_l]}(z^{[\kappa_l]})$,where $\kappa_l \geq 0$,...
18: and dummy derivatives for $z_l^{[\kappa_l]}$, where $\kappa_l > 0$ and original variables,...
19: as well as all equations $g_l^{(\check{\mathbf{c}})}, \ldots, g_l^{(\mathbf{c})}$,...
20: and dummy derivatives for all variables $x_j^{(\check{d}_j)}, \ldots, x_j^{(d_j)}$.

discussion we will call DDs *block necessary dummy derivatives* if they are found at the end of Algorithm 3, i.e. if they can be found by using a block form's local offsets and the global canonical offsets. There is no explicit interaction between blocks in Algorithm 3 (the inter block dependencies are taken in to account on lines 19 and 20) and therefore the majority of the algorithm could be performed in parallel. In our discussions that follow we will restrict ourselves to thinking about coarse and fine BTFs, as described in Sect. 2.3, as these tend to be the most natural when using the signature matrix method.

To prove Algorithm 3 gives a suitable choice of DDs we would like an analogue of a Theorem in [20] that asserts the difference between local and global offsets is constant

on a fine block form to hold for an arbitrary block form. This would mean that we introduce only as many DDs as differentiated equations when taking the interactions between blocks in to account. Consider however the following example:

Example 10 Consider a DAE with the following signature matrix:

$$\begin{array}{cccc|cc} & & & & c_i & \check{c}_i \\ \hline \multicolumn{1}{|c}{0} & 1 & & \multicolumn{1}{c|}{} & 0 & 0 \\ \hline \multicolumn{1}{|c}{} & 0 & 0 & \multicolumn{1}{c|}{} & 1 & 0 \\ \multicolumn{1}{|c}{} & 0 & 1 & \multicolumn{1}{c|}{} & 0 & 0 \\ \hline d_j & 0 & 1 & 1 & & \\ \check{d}_j & 0 & 0 & 1 & & \end{array} .$$

Here we are block triangularising over coarse blocks. There is not a constant difference between coarse local and global offsets, so it's possible we introduce more DDs than we do equations when using Algorithm 3.

The potential problem in Example 10 is actually not a problem at all:

Theorem 6 *Given a BTF of Σ the difference between the sum of any block's local offsets and global offsets is equal with respect to* $\mathbf{c}$ *and* $\mathbf{d}$.

Proof Take an $n \times n$ signature matrix Σ and put its HVT on the main diagonal. Let S be any subset of $\{1, \ldots, n\}$ and $\check{\mathbf{c}}$ and $\check{\mathbf{d}}$ be any valid offsets. Then, since $\check{d}_i - \check{c}_i = \sigma_{i,i}$ we have:

$$\sum_{i \in S} \check{d}_i - \sum_{i \in S} \check{c}_i = \sum_{i \in S} \sigma_{i,i} = \mathrm{Val}(S)$$

which is independent of $\check{\mathbf{c}}$ and $\check{\mathbf{d}}$. So, for any other valid offsets, say $\mathbf{c}$ and $\mathbf{d}$ we have:

$$\sum_{i \in S} \check{d}_i - \sum_{i \in S} d_i = \sum_{i \in S} \check{c}_i - \sum_{i \in S} c_i = \mathrm{Val}(S)$$

□

This is not what one might first expect: in our coarse block algorithm one might expect to differentiate an entire block a number of times to solve a later block, this Theorem shows that actually you may only need to differentiate some parts of the block to retain a square index 1 system using lines 19 and 20 in Algorithm 3.

Theorem 7 *Algorithm 3 gives a suitable choice of DDs that could otherwise have been found by considering the entire system and original DD algorithm.*

Proof If one is able to do the above algorithm then we have a block form whose diagonal sub-matrices are structurally non-singular. Because each block's coarse local offsets are a constant away from the global, by Lemma 1 we must have a non-singular Jacobian at each global stage of DDs if we have a non-singular Jacobian at each coarse local stage. Again, by Lemma 1 we see that a valid choice for the global stage DD

algorithm is an amalgam of variables found at positive and negative local stages, using each block's lead times (if we consider adding variables and equations at the end of the algorithm as doing local stages $0, \ldots, \max_j (d_j - \check{d}_j)$). □

Due to Theorem 5 and Example 7 one may think when restricting to coarse blocks Algorithm 3 is just a restating of Algorithm 2—consider a DAE with signature matrix:

	x_1	x_2	x_3	x_4	x_5	x_6	c_i
f_1	5	0					0
f_2		0	4				0
f_3	0		0			0	4
f_4				0	0		6
f_5					2	0	4
f_6				2		0	4
d_j	5	0	4	6	6	4	

k	-6	-5	-4	-3	-2	-1
m_k	1	1	4	4	4	4
n_k	2	3	5	5	5	5

Looking for stages where $m_k = n_k$ we find no structural DDs. However, using Algorithm 3 we find block necessary DDs for $x_4^{(3)}, \ldots, x_4^{(6)}, x_5^{(3)}, \ldots, x_5^{(6)}$ and $\dot{x}_6, \ldots, x_6^{(4)}$.

We see that Algorithm 3 restricted to the coarse BTF yields a valid set of DDs that could be found using the global offsets and no BTF. All we've done is take a BTF, so one might assume in general it generates all possible sets of valid DDs that could be obtained globally, as certainly seems to be asserted in the original paper [7]. Consider however:

Example 11 Consider a DAE with signature matrix as follows:

$$\Sigma = \begin{array}{c|ccccc|c} & x_1 & x_2 & x_3 & x_4 & x_5 & c_i \\ f_1 & 1 & 0 & & & 0 & 1 \\ f_2 & 2 & 1 & & & & 0 \\ \hline f_3 & & & 1 & 0 & & 2 \\ f_4 & & & 2 & 1 & 0 & 1 \\ f_5 & & & & 2 & 1 & 0 \\ \hline d_j & 2 & 1 & 3 & 2 & 1 & \end{array}$$

Note that the coarse BTF given above is the same as the fine BTF and $\mathbf{c} = \check{\mathbf{c}} = \hat{\mathbf{c}}$ and $\mathbf{d} = \check{\mathbf{d}} = \hat{\mathbf{d}}$. Algorithm 3 will give a DD for either $\ddot{x}_1$ or $\dot{x}_2$ from the first block, that is Algorithm 3 cannot find a set of DDs where neither of $\ddot{x}_1$ and $\dot{x}_2$ are selected. However, if we carry out the original DD algorithm we see that there is an index one system that uses only derivatives of variables x_3, x_4 and x_5 as DDs. Thus there are index 1 systems we may 'miss' using a BTF that would otherwise be available if considering the DAE as a whole.

4.5 Fine block based dummy derivatives

We now show that the choice of block form does directly affect the number of block necessary DDs as defined in Sect. 4.4. We assert the fine BTF is a good choice. Before giving an example of Algorithm 3 using the fine BTF we wonder if there exists a more informative version of Algorithm 2 based on the fine BTF. Consider again Eq. (8), its signature matrices in Fig. 1 and its stages in Table 5. For stages -2 and -1 $m_k \neq n_k$ because we have to introduce initial values for $\ddot{x}_5$ and $\dot{x}_5$. If we did not have to do this we would again have a square system, this time with 4 equations—the original coarse block containing equations 1, 2, 3 and variables 1, 2, 3 and a fine block containing equation 6 and variable 4. We note that variable 5 does not actually appear in the DD solution scheme until stage 0 because it does not have an associated equation. Therefore we can ignore any variables that must be specified as IVs and are left with the following definition

Definition 4 Fine block local numbers of equations and variables are written as:

$$\widehat{m}_{k,l} = \left|\{j \text{ in block } l \mid \widehat{d_j} + k \geq 0\}\right|, \quad \widehat{n}_{k,l} = \left|\{i \text{ in block } l \mid \widehat{c_i} + k \geq 0\}\right|$$

where k is taken to be a local SA stage associated with a fine block.

Doing this we end up with an improved fine block based analogue of Algorithm 2. For each fine block l do the following (where $\widehat{k}_{\min,l}$ is the $-\max_j d_j$ such that j is in block l):

Algorithm 4 Fine BTF based structural DD algorithm

1: **for** $K = \widehat{k}_{\min,l} : -1$ **do**
2: Find SA solution scheme
3: Note stages where $\widehat{m}_{k,l} = \widehat{n}_{k,l}$
4: Make subsequent derivatives of such variables DDs
5: **for** $K = 0 : -\widehat{k}_{\min,l}$ **do**
6: Work through DDs algorithm, but:
7: keep columns relating to already known DDs from step 4 when finding $G^{[K]}$

Knowing this it motivates us to continue using the fine BTF to find DDs, since we have a better set of structural DDs (we will term such DDs block structurally necessary DDs) than was previously found using the entire signature matrix.

Example 12 Consider again the DAE in Example 4. We carry out Algorithm 3 on this DAE to illustrate the advantages of the fine BTF. The local offsets tell us that we must have DDs shown in Table 6—found by comparing the offsets via lines 19 and 20 of the Algorithm.

The only block we have to select DDs in is block 4 (equivalent to the simple pendulum). For ease of checking the Jacobians that follow we now consider this as a stand alone DAE (as would be done should Algorithm 3 be done in parallel over fine blocks) and give the differentiated problem $\mathcal{G}x = 0$:

Table 6 DDs from the fine blocks for example (8)

Block number	Fine block structural DDs
1	No dummy derivatives for this block
2	No dummy derivatives for this block
3	$\dot{x}_4, \ddot{x}_4$
4	$x_1^{(3)}, x_1^{(4)}, x_1^{(5)}, x_1^{(6)}, \quad x_2^{(3)}, x_2^{(4)}, x_2^{(5)}, x_2^{(6)}, \quad \dot{x}_3, \ddot{x}_3, x_3^{(3)}, x_3^{(4)}$

Table 7 Possible DD choices from (8)

Stage 1	Stage 2
x_1''	x_1'
x_2''	x_2'

$$
\left.\begin{aligned} f_1 &= \ddot{x}_1 + x_1 x_3 &&= 0 \\ f_2 &= \ddot{x}_2 + x_2 x_3 - G &&= 0 \\ f_3 &= x_1^2 + x_2^2 - L^2 &&= 0 \end{aligned}\right\}
$$

$$
\left.\begin{aligned} \dot{f}_3 &= f_4 = 2\dot{x}_1 x_1 + 2\dot{x}_2 x_2 &&= 0 \\ \ddot{f}_3 &= f_5 = 2\ddot{x}_1 x_1 + 2\dot{x}_1^2 + 2\ddot{x}_2 x_2 + 2\dot{x}_2^2 &&= 0 \end{aligned}\right\}
$$

Performing the DD algorithm gives two possible index 1 systems, where the choice of DDs is given in Table 7. We are able to find 14 block necessary DDs without ever having to compute a numerical Jacobian and reduced our problem size by half at the outset.

A brief complexity analysis follows. Assume the DAE decomposes into L fine blocks labelled by a subscript l. At each stage of DDs the original algorithm has a complexity of order:

$$
\sum_{k=-1}^{k_{\min}} (n^k)^3
$$

because the selection of DDs is usually done via a QR decomposition in practice, which is an $\mathcal{O}(n^3)$ operation. The proposed algorithm has a complexity of order:

$$
\sum_{l=1}^{L} \left(\sum_{k=-1}^{\widehat{k}_{\min,l}} (\widehat{n}_{k,l})^3 \right)
$$

where $\widehat{c_i}$ in the second summation is taken to be in block l. Assuming the system decomposes into relatively small fine blocks, i.e. $\widehat{n} \ll n$ and some $\widehat{c_i} < c_i$—from test models in DAESA this is usually the case—this should offer good numerical speed up.

Importantly the large reduction in potential DDs and problem size makes dummy pivoting less problematic, since we will have to consider smaller systems of equations with a reduced number of potential variables to choose from, see Sect. 5 for more detail.

Consider again Example 11, we are reminded that there are index 1 systems we may 'miss' using a fine BTF that would otherwise be available if considering the DAE as a whole. Whilst this seems like a rather large oversight of our method, this potential 'oversight' will never make an otherwise solvable (by DDs) DAE unsolvable:

Theorem 8 *If the DAE is solvable then Algorithm 3 restricted to the fine BTF will always be able to select a valid set of DDs.*

Proof If the system is solvable then there must exist DD matrices globally, i.e. there is a choice $G^{[0]}, \ldots, G^{[\max_i c_i]}$ for which each matrix is non-singular. We also know that there exist global SA system Jacobian $\mathbf{J}_0, \ldots, \mathbf{J}_{-\max_i c_i}$ that have full row rank. If the SA scheme succeeds globally then it succeeds over fine-blocks. That is, over each fine block l there are $\mathbf{J}_{0,l}, \ldots, \mathbf{J}_{-\max_i \hat{c}_i,l}$ that have full row rank. Since we have that any $\mathbf{J}_0 = G^{[0]}$ and subsequent DD matrices are sub matrices of previous SA matrices it must be possible to select a non-singular DD matrix on each fine block if each SA Jacobian has full row rank in that block. □

Thus, although it is possible to find non-singular choices of DDs globally that cannot be found over a fine block, these choices are somehow a redundant selection—although admittedly it might be possible the global selections have better condition numbers there will exist a fine block selection if the DAE is solvable.

5 Dummy pivoting

We briefly discuss the issue of dummy pivoting as first examined in [8]. When proceeding along a numerical solution it is possible that some chosen matrices $G^{[\kappa]}$ will become singular. It is therefore necessary to change our choice of $G^{[\kappa]}$ as our numerical solution evolves with time. Such a change is called dummy pivoting or dynamic selection of states [9]. The main problem with making such a change is that changing a $G^{[\kappa]}$ will (frequently, but not necessarily) produce a need to change all subsequent $G^{[\kappa+i]}$. This problem is made worse when the DAE is of larger index (there will be multiple stages and thus more potential $G^{[\kappa]}$ to change) or when the problem is large (this will result in estimating condition numbers of potentially large matrices frequently throughout the solution process). One method of avoiding the former problem is to start by attempting to reduce the order of the problem and block triangularising as in Sect. 4.5, a technique that in practice frequently reduces the order. Another alternative is to select which states must or cannot be chosen as DDs, this is implemented in some tools, such as is done in DYMOLA with each variable given a `stateSelect` parameter that can be `prefer`, `default` or `avoid`. Selecting dynamic states before solving however needs to be informed by problem specific knowledge, which it is not always possible a user will have. It is also dangerous as the user may avoid choosing necessary DDs if they are not careful. We offer an insight into the number of different potential Jacobians $G^{[\kappa]}$ gained by the following example.

Example 13

$$\left.\begin{aligned} f_1 &= \ddot{x}_1^2 + \dot{x}_5^2 + u_1(t) &&= 0 \\ f_2 &= \ddot{x}_4^2 + \ddot{x}_2^2 + u_2(t) &&= 0 \\ f_3 &= \dot{x}_1^2 + x_3^2 + u_3(t) &&= 0 \\ f_4 &= x_4^2 + x_3^2 + u_4(t) &&= 0 \\ f_5 &= \dot{x}_5^2 + \dot{x}_2^2 + u_5(t) &&= 0 \end{aligned}\right\}. \tag{13}$$

Here $u_1(t), \ldots, u_5(t)$ are arbitrary driving functions. The non-linearity gives stages in the DD algorithm that may need pivoting. This DAE (13) has signature matrix and offsets:

$$G^{[0]} = \begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ d_j \end{array} \begin{array}{c} \begin{array}{ccccc} x_1 & x_2 & x_3 & x_4 & x_5 \end{array} \\ \begin{pmatrix} 2^{\bullet} & & & & 1^{\circ} \\ & 2^{\bullet} & & 2^{\circ} & \\ 1^{\circ} & & 0^{\bullet} & & \\ & & 0^{\circ} & 0^{\bullet} & \\ & 1^{\circ} & & & 1^{\bullet} \end{pmatrix} \\ \begin{array}{ccccc} 3 & 2 & 2 & 2 & 2 \end{array} \end{array} \begin{array}{c} c_i \\ 1 \\ 0 \\ 2 \\ 2 \\ 1 \end{array}$$

This is irreducible and has two HVTS, marked by $\bullet$ and $\circ$. Every entry of Σ corresponds to a structurally non-singular entry in each $\mathbf{J}_k$ and their related DD stage Jacobians. For ease of checking entries in the DD stage Jacobians that follow we present all equations and their derivatives specified by $\mathbf{c}$:

$$\left.\begin{aligned} f_1 &= \ddot{x}_1^2 + \dot{x}_5^2 + u_1(t) &&= 0 \\ f_2 &= \ddot{x}_4^2 + \ddot{x}_2^2 + u_2(t) &&= 0 \\ f_3 &= \dot{x}_1^2 + x_3^2 + u_3(t) &&= 0 \\ f_4 &= x_4^2 + x_3^2 + u_4(t) &&= 0 \\ f_5 &= \dot{x}_5^2 + \dot{x}_2^2 + u_5(t) &&= 0 \end{aligned}\right\}$$

$$\left.\begin{aligned} \dot{f}_1 = f_6 &= 2x_1^{(3)}\ddot{x}_1 + 2\ddot{x}_5\dot{x}_5 + \dot{u}_1(t) &&= 0 \\ \dot{f}_3 = f_7 &= 2\ddot{x}_1\dot{x}_1 + 2\dot{x}_3x_3 + \dot{u}_3(t) &&= 0 \\ \dot{f}_4 = f_8 &= 2\dot{x}_4x_4 + 2\dot{x}_3x_3 + \dot{u}_4(t) &&= 0 \\ \dot{f}_5 = f_9 &= 2\ddot{x}_5\dot{x}_5 + 2\ddot{x}_2\dot{x}_2 + \dot{u}_5(t) &&= 0 \end{aligned}\right\}$$

$$\left.\begin{aligned} \ddot{f}_3 = f_{10} &= 2x_1^{(3)}\dot{x}_1 + 2\ddot{x}_1^2 + 2\ddot{x}_3x_3 + 2\dot{x}_3^2 + \ddot{u}_3(t) &&= 0 \\ \ddot{f}_4 = f_{11} &= 2\ddot{x}_4x_4 + 2\dot{x}_4^2 + 2\ddot{x}_3x_3 + 2\dot{x}_3^2 + \ddot{u}_4(t) &&= 0 \end{aligned}\right\}.$$

Stage 0: In the DD algorithm, we have an initial Jacobians:

$$G^{[0]} = \begin{array}{c} \\ \dot{f}_1 \\ f_2 \\ \dot{f}_3 \\ \ddot{f}_4 \\ \dot{f}_5 \\ d_j \end{array} \begin{array}{c} \begin{array}{ccccc} x_1^{(3)} & \ddot{x}_2 & \ddot{x}_3 & \ddot{x}_4 & \ddot{x}_5 \end{array} \\ \begin{pmatrix} 2\ddot{x}_1 & 0 & 0 & 0 & 2\dot{x}_5 \\ 0 & 2\ddot{x}_2 & 0 & 2\ddot{x}_4 & 0 \\ 2\dot{x}_1 & 0 & 2x_3 & 0 & 0 \\ 0 & 0 & 2x_3 & 2x_4 & 0 \\ 0 & 2\dot{x}_2 & 0 & 0 & 2\dot{x}_5 \end{pmatrix} \\ \begin{array}{ccccc} 3 & 2 & 2 & 2 & 2 \end{array} \end{array} \begin{array}{c} c_i \\ 1 \\ 0 \\ 2 \\ 2 \\ 1 \end{array},$$

$$H^{[0]} = \begin{array}{c} \\ \dot{f}_1 \\ \ddot{f}_3 \\ \ddot{f}_4 \\ \dot{f}_5 \end{array} \begin{array}{c} \begin{array}{ccccc} x_1^{(3)} & \ddot{x}_2 & \ddot{x}_3 & \ddot{x}_4 & \ddot{x}_5 \end{array} \\ \begin{pmatrix} 2\ddot{x}_1 & 0 & 0 & 0 & 2\dot{x}_5 \\ 2\dot{x}_1 & 0 & 2x_3 & 0 & 0 \\ 0 & 0 & 2x_3 & 2x_4 & 0 \\ 0 & 2\dot{x}_2 & 0 & 0 & 2\dot{x}_5 \end{pmatrix} \end{array}.$$

Stage 1: We need to find $G^{[1]}$. We do this by choosing any 4 columns, since all 5 possibilities are (at least structurally) non-singular. Numerically one potentially has to pivot between different systems, for example say you chose columns 1, 2, 3, 4 and $\dot{x}_2 \to 0$ at some time t then you would pivot to the system with columns 1, 3, 4, 5. Choose columns 1, 2, 3, 4 giving DDs for $x_1^{(3)}$, $\ddot{x}_2$, $\ddot{x}_3$ and $\ddot{x}_4$. The new Jacobians are:

$$G^{[1]} = \begin{array}{c} \\ f_1 \\ \dot{f}_3 \\ \dot{f}_4 \\ f_5 \end{array} \begin{array}{c} \begin{array}{cccc} \ddot{x}_1 & \dot{x}_2 & \dot{x}_3 & \dot{x}_4 \end{array} \\ \begin{pmatrix} 2x_1'' & 0 & 0 & 0 \\ 2\dot{x}_1 & 0 & 2x_3 & 0 \\ 0 & 0 & 2x_3 & 2x_4 \\ 0 & 2\dot{x}_2 & 0 & 0 \end{pmatrix} \end{array}. \quad \text{and}$$

$$H^{[1]} = \begin{array}{c} \\ \dot{f}_3 \\ \dot{f}_4 \end{array} \begin{array}{c} \begin{array}{cccc} \ddot{x}_1 & \dot{x}_2 & \dot{x}_3 & \dot{x}_4 \end{array} \\ \begin{pmatrix} 2\dot{x}_1 & 0 & 2x_3 & 0 \\ 0 & 0 & 2x_3 & 2x_4 \end{pmatrix} \end{array}.$$

Stage 2: Choose two columns from $H^{[1]}$ to form a square non-singular matrix. There are three choices. As above it may be necessary to pivot chosen DDs numerically. Say we choose columns 1 and 3, then we introduce DDs for $\ddot{x}_1$ and $\dot{x}_3$ and get the following Jacobian:

$$G^{[2]} = \begin{array}{c} \\ f_3 \\ f_4 \end{array} \begin{array}{c} \begin{array}{cc} \dot{x}_1 & x_3 \end{array} \\ \begin{pmatrix} 2\dot{x}_1 & 2x_3 \\ 0 & 2x_3 \end{pmatrix} \end{array}.$$

Removing undifferentiated equations yields $H^{[2]} = [\,]$ and the algorithm terminates. Note: we have selected the following dummies: x_1'', $x_1^{(3)}$, x_2'', x_3', x_3'' and x_4''. If we

Table 8 Possible DD index 1 systems from (13)

Stage 1	Stage 2
$x_1^{(3)}, x_2'', x_3'', x_4''$	x_1'', x_3' or x_1'', x_4' or x_3', x_4'
$x_1^{(3)}, x_2'', x_3'', x_5''$	x_1'', x_3'
$x_1^{(3)}, x_2'', x_4'', x_5''$	x_1'', x_4'
$x_1^{(3)}, x_3'', x_4'', x_5''$	x_1'', x_3' or x_1'', x_4' or x_3', x_4'
$x_2'', x_3'', x_4'', x_5''$	x_3', x_4'

do not check for non-singular matrices at each stage we would have $\binom{5}{4}\binom{4}{2} = 30$ possible index 1 systems at the end of the algorithm. If however we check for structural singularity at each stage (as is done above) we get 9 possible index 1 systems. Listed in Table 8 are selected DDs.

If one thinks of the number of possible DD index 1 systems as a tree diagram, with stage 1 producing the root and each subsequent stage producing branching nodes it is possible to make pivoting easier. In applications where the DAE is usually of large size but (relatively) low index the tree will likely be wide but shallow. Pivoting can then be done across nodes on each level, i.e. you can pivot between all nodes at one level (restricting yourself to nodes coming from one parent node), starting at the node that gives you a singular matrix $G^{[\kappa]}$ and considering its leaves, then if no nodes at the current level give a non singular matrix go up a level and repeat. Storing such a diagram may take a lot of memory, but it's highly likely many nodes will in fact have the same $G^{[\kappa]}$. Consider Table 8. We see there are actually only three distinct nodes at stage 2, so we only have to store 3 Jacobians, not 9 and check 3 Jacobians for non-singularity in the worst case.

6 Conclusions

In this paper we have outlined the Signature matrix method of [18] in Sect. 2 and a reordered version of the Dummy Derivative method of [7] in Sect. 3 to draw parallels between the approaches and develop a method of doing DDs that is structurally well informed. The concept of structurally necessary DDs has been defined and explored for different block forms in Sect. 4. Such analysis should help those using the DD algorithm for index reduction to better understand the underlying restrictions on their reformulated index 1 system. We have also expanded upon the original description of block based DDs and demonstrated that one can restrict their potential index 1 system choices by using a block form, although such a restriction will not make a DAE unsolvable if it was otherwise solvable. We believe such a rigorous analysis of DDs using SA as a base will help modellers understand the subtleties inherent in this index reduction procedure and give them some insights when developing their models. We believe the complexity analysis for our revised algorithm will drive modellers to use it on their problems where they were previously using a different index reduction

procedure. It is our opinion that dynamic selection of states is not a widely understood topic in the DAE community. We hope that Sect. 5 is useful to modellers that have to incorporate dummy pivoting into their codes.

References

1. Brenan, K., Campbell, S., Petzold, L.: Numerical solution of initial-value problems in differential-algebraic equations. SIAM, Philadelphia (1996)
2. Campbell, S., Griepentrog, E.: Solvability of general differential algebraic equations. SIAM J. Sci. Comput. **16**(2), 257–270 (1995)
3. Dynasym AB, Dymola Dynamic Modeling Laboratory, User's Manual, http://www.inf.ethz.ch/personal/cellier/Lect/MMPS/Refs/Dymola5Manual (2004)
4. Fritzson, P.: Principles of object-oriented modeling and simulation with Modelica 3.3: a cyber-physical approach. wiley-ieee press, Hoboken (2015)
5. Griewank, A.: Evaluating derivatives: principles and techniques of algorithmic differentiation. SIAM, Philidelphia (2000)
6. Hindmarsh, A., Brown, P., Grant, K., Lee, S., Serban, R., Shumaker, D., Woodward, C.: SUNDIALS: suite of nonlinear and differential/algebraic equation solvers. ACM Trans. Math. Softw. **31**(3), 363–396 (2005)
7. Mattsson, S., Söderlind, G.: A New Technique for Solving High-Index Differential-Algebraic Equations Using Dummy Derivatives, Computer-Aided Control System Design (1992)
8. Mattsson, S., Söderlind, G.: Index reduction in differential-algebraic equations using dummy derivatives. SIAM J. Sci. Comput. **14**, 677–692 (1993)
9. Mattsson, S., Olsson, H., Elmqvist, H.: dynamic selection of states in DYMOLA (2000)
10. McKenzie, R., Nedialkov, N., Pryce, J., Tan, G.: DAESA User Guide, Department of Computing and Software. McMaster University (2013)
11. McKenzie, R.: Exploiting underlying DAE structures using the Signature Matrix method for index reduction (PhD Thesis), Department of Mathematics, Cardiff University (2016)
12. Nedialkov, N., Pryce, J.: Solving differential-algebraic equations by Taylor series (I): computing Taylor coefficients. BIT **45**, 561–591 (2005)
13. Nedialkov, N., Pryce, J.: Solving differential-algebraic equations by Taylor series (III): the DAETS code. JNAIAM **3**, 61–80 (2007)
14. Pantelides, C.: The consistent initialization of differential-algebraic systems. SIAM. J. Sci. Stat. Comput. **9**, 213–231 (1988)
15. Petzold, L.: Description of DASSL: a differential/algebraic system solver. http://www.osti.gov/scitech/servlets/purl/5882821 (1982)
16. Process Systems Enterprise Ltd., gPROMS Introductory User Guide. http://eng1.jcu.edu.au/Current
17. Pryce, J.: Solving High-Index DAEs by Taylor Series. Numer. Algorithms **19**, 195–211 (1998)
18. Pryce, J.: A simple structural analysis method for DAEs. BIT **41**(2), 364–394 (2001)
19. Pryce, J., Nedialkov, N., Tan, G.: Graph theory, irreducibility, and structural analysis of differential-algebraic equation systems, Cardiff University. McMaster University. In: preparation (2015)
20. Pryce, J., Nedialkov, N., Tan, G.: DAESA—a matlab tool for structural analysis of differential-algebraic equations: theory (2015). ACM Trans. Math. Softw. **41**(2), 1–20 (2015)
21. Washington, I., Swartz, C.: On the numerical robustness of differential-algebraic distillation models, 61st Canadian Chemical Engineering Conference. Ontario, Canada, London (2011)