A

DAESA — a Matlab Tool for Structural Analysis of Differential-Algebraic Equations: Theory

JOHN D. PRYCE, Cardiff University
NEDIALKO S. NEDIALKOV, McMaster University
GUANGNING TAN, McMaster University

DAESA, Differential-Algebraic Equations Structural Analyzer, is a MATLAB tool for structural analysis of differential-algebraic equations (DAEs). It allows convenient translation of a DAE system into MATLAB and provides a small set of easy-to-use functions. DAESA can analyze systems that are fully nonlinear, high-index, and of any order. It determines structural index, number of degrees of freedom, constraints, variables to be initialized, and suggests a solution scheme. The structure of a DAE can be readily visualized by this tool. It also can construct a block-triangular form of the DAE, which can be exploited to solve it efficiently in a block-wise manner.

This paper describes the theory and algorithms underlying the code.

Categories and Subject Descriptors: I.6.7 [**Computing methodologies**]: Simulation support systems; G.4 [**Mathematical software**]: Matlab, Algorithm design and analysis

Additional Key Words and Phrases: Differential-algebraic equations, structural analysis, modeling

ACM Reference Format:
Pryce, J., Nedialkov, N., Tan, G., 2012. DAESA — A Matlab Tool for Structural Analysis of DAEs: Theory ACM Trans. Math. Softw. V, N, Article A (YYYY), 20 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

For some years the authors have been developing a numerical code DAETS, see Nedialkov and Pryce [2008; 2009], for solving differential-algebraic equation systems (DAEs). It is based on a structural analysis (SA) of the sparsity of the DAE that we call the *signature matrix method*, or Σ-*method*. To a large extent, it is equivalent to the well-known method of Pantelides [1988], and in particular computes the same *structural index* [Duff and Gear 1986; Pryce 2001]. However, our method is easier to apply and can be applied to DAEs of any order, not just first order.

Large DAE systems are produced routinely by equation-based modeling methods in disciplines such as electronic circuits, the study of robots and other mechanical systems, chemical engineering, etc. It is now routine that such models are built using interactive design systems (GPROMS, MAPLESIM, SIMULINK, various tools based on the MODELICA language, etc. [Cameron and Gani 2011]). Mostly these have some kind of SA built in.

For instance, GPROMS [Process Systems Enterprise Ltd. 2004] uses the method of Pantelides [1988] to determine if the DAE is of index 1. If so, GPROMS checks if

Author's addresses: J. Pryce, Cardiff School of Mathematics, Cardiff University, UK. N. Nedialkov and G. Tan, Department of Computing and Software, McMaster University, Hamilton, Ontario, Canada.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

the given initial values are consistent, and if they are, integrates the problem using DASSL [Brenan et al. 1996]. In the case when the problem is not well posed, GPROMS detects over-specified and under-specified parts and provides diagnostic information to the user. If the DAE is of index greater than 1, GPROMS reports a subset of equations and variables that cause the higher index.

DYMOLA [Dynasym AB 2004] also uses Pantelides's algorithm to determine the index of a DAE and then applies the dummy-derivative index reduction technique [Mattsson and Söderlind 1993] to convert it to an index-1 problem, which is then solved by DASSL.

Originally, our SA was merely a necessary preprocessing stage to set up the numerical solution method for a DAE initial value problem (IVP). However, as we have encountered users with increasingly large problems, it has become clear that they value its diagnostic abilities, not all which are present in systems such as GPROMS and DYMOLA mentioned above, although there is a large overlap. In particular, our SA is able to identify subsystems of a DAE, and the hierarchy of dependencies among them, to a finer resolution than many other methods. Thereby, it can often reduce the number of initial values (IVs) required for numerical solution, beyond what those other methods achieve.

Thus it seemed useful to present it as a free-standing tool, with enhanced reporting and diagnostic capabilities. The result is the program DAESA. Written in MATLAB, it accepts a MATLAB description of a DAE, which is nearly identical to the C++ description accepted by DAETS. The theory behind DAESA is presented here, and its current facilities are described in detail by the companion paper [Nedialkov et al. 2012].

Section 2 states the class of problem DAESA handles and describes the basic theory of the DAE's *linear assignment problem* and *offsets*, leading to a stage-wise solution scheme. Section 3 discusses consistent points and introduces the problem of minimizing the number of IVs the user must supply. Section 4 describes different block-triangular forms for the DAE and how they may simplify numerical solution and reduce the number of IVs. Section 5 presents examples illustrating the previous sections. Section 6 discusses some implementation issues. Finally, Section 7 summarizes the facilities DAESA offers and some ideas for future development. Small examples in the text illustrate the theory; larger examples are in the companion paper.

2. OVERVIEW OF THE SIGNATURE-MATRIX METHOD

We present the class of problems DAESA handles (§2.1), describe how we compute the *offsets* of the problem, (§2.2), and outline the solution scheme based on the Σ-method (§2.3). For the basic ideas and results summarized here, see [Pryce 2001] unless stated otherwise.

2.1. The class of DAE handled by DAESA

The code DAETS solves DAE IVPs by expansion in Taylor series. DAESA is an offshoot of it, and performs essentially the same SA. Both codes handle DAEs of the general form

$$f_i(\,t,\text{ the } x_j \text{ and derivatives of them}\,) = 0, \quad i = 1, \ldots, n, \tag{1}$$

where the $x_j(t),\ j = 1, \ldots, n$ are state variables that are functions of an independent (time) variable t. The f_i can be arbitrary expressions built from the x_j and t using $+, -, \times, \div$, other analytic standard functions, and the differentiation operator d^p/dt^p. They can be nonlinear and fully implicit in the variables and derivatives. An equation such as

$$\left((tx_1')'\right)^2 / \left(1 + (x_2'')^2\right) + t^2 \cos x_2 = 0$$

can be encoded directly into either code.

We call our approach the Σ-method, because it is based on the $n \times n$ *signature matrix* Σ, whose i, j entry σ_{ij} is either an integer ≥ 0, namely the order of the highest derivative to which variable x_j occurs in the function f_i; or $-\infty$ if x_j does not occur in f_i. This compact description provably represents the essential structure for several classical DAE forms, such as Hessenberg (block-nearly-triangular). Perhaps unexpectedly, it does so also for a large number of DAEs that occur in applications and do not obviously fall in one of the classical forms.

2.2. The linear assignment problem

The start of the process is to take Σ as the matrix of a *linear assignment problem* (LAP). Let the variables (columns) represent n workers and the equations (rows) represent n tasks. Then σ_{ij} represents the competence of worker i at doing task j, with $-\infty$ meaning total incompetence. The problem is to assign one worker per task to maximize the competence of the team, measured as the sum of individual competences. Each such assignment can be specified by a *transversal*, a set T of n positions (i, j), with just one entry in each row and each column; the team competence is then the sum $\sum_{(i,j)\in T} \sigma_{ij}$, which we call the *value* of T, written $\mathrm{Val}(T)$.

We seek a *highest-value transversal*, or HVT, that gives $\mathrm{Val}(T)$ its largest possible value: this is called the *value of the signature matrix*, $\mathrm{Val}(\Sigma)$. The DAE is *structurally well-posed*, if it has a T, all of whose σ_{ij} are finite (so $\mathrm{Val}(T)$ and hence $\mathrm{Val}(\Sigma)$ are finite), else structurally ill-posed.

A LAP is a kind of linear programming problem (LPP), so it has a dual problem. In the formulation we use, this has $2n$ dual variables, $c = (c_1, \ldots, c_n)$ and $d = (d_1, \ldots, d_n)$, associated with the equations and the variables of (1), respectively. The dual LPP consists of minimizing $\sum d_j - \sum c_i$ subject to[1]

$$d_j - c_i \geq \sigma_{ij} \text{ for all } i, j, \tag{2}$$

together with $c_i \geq 0$ for all i.

Assume henceforth a structurally well-posed DAE. Then both the primal and the dual LPP have feasible solutions, so the two objective functions have the same optimal value, which is $\mathrm{Val}(\Sigma)$. When the method succeeds (see §3.2), this equals the *number of degrees of freedom* (DOF) of the DAE.

Any optimal solution vectors c, d of the dual are termed *valid offsets*. They have $d_j \geq 0$ as well as $c_i \geq 0$, and are characterized by

$$c_i \geq 0; \quad d_j - c_i \geq \sigma_{ij} \text{ for all } i, j, \text{ with equality on some HVT, hence on all HVTs,} \tag{3}$$

see [Pryce 2001, Theorem 3.4]. Valid offsets are never unique, e.g., if c_i, d_j solve (3), then so do $c'_i = c_i + K,\ d'_j = d_j + K$ for any constant K provided $c'_i \geq 0$, i.e. $K \geq -\min_i c_i$. However [ibid., Theorem 3.6], there exists a unique elementwise smallest solution of (3) called the *canonical offsets*. It is convenient, but not essential, to base the SA and numerical solution on these.

[1] Historically, the inequalities (2) were identified first as key to the solution process, and the LAP derived from them.

Example 2.1. The DAE of the simple pendulum in Cartesian coordinates will be used frequently as an example:

$$\begin{array}{c}\textsc{Pend}\\ 0=A=x''+x\lambda\\ 0=B=y''+y\lambda-G\\ 0=C=x^2+y^2-L^2\end{array}\qquad \Sigma=\begin{array}{c|ccc|c} & x & y & \lambda & c_i\\ A & 2^\bullet & & 0^\circ & 0\\ B & & 2^\circ & 0^\bullet & 0\\ C & 0^\circ & 0^\bullet & & 2\\ \hline d_j & 2 & 2 & 0 & \end{array} \tag{4}$$

x, y and λ are state variables. G and L are constants >0. A blank in Σ means $-\infty$.

The rows and columns of the signature matrix are annotated on the left and top with the names of the functions A, B, C and the variables x, y, λ, and on the right and bottom with the offsets. To match the general notation of (1), the functions and variables are regarded as also being named f_1, f_2, f_3 and x_1, x_2, x_3 respectively, in the order given. There are two HVTs here, marked with $\bullet$ and $\circ$.

Remark 2.2. LAPs are solved in DAESA using Y. Cao's MATLAB implementation [Cao 2011] of the Jonker-Volgenant algorithm [Jonker and Volgenant 1987].

2.3. The basic staged solution scheme

A basic solution approach, in theory and practice, is to differentiate equations one or more times (with respect to t) to obtain an enlarged system that can be solved to give an ODE in the x_j. The smallest number of differentiations of any equation, needed to do this, is the DAE's *differentiation index* ν.

The offsets produced by the Σ-method prescribe a stage-by-stage solution process. Viewed symbolically, this comprises a sequence of differentiations and applications of the Implicit Function Theorem that convert the DAE to an ODE; or, if not carried to completion, convert it to one of lower index (*index-reduction*). Viewed numerically, it yields various methods, e.g., the one currently used by DAETS, which expands the solution in Taylor series at each integration step.

The number of differentiations specified by this staged process gives a formula for the *structural index* ν_S of the DAE:

$$\nu_S=\max_i c_i+\begin{cases}1 & \text{if some } d_j \text{ is zero,}\\ 0 & \text{otherwise.}\end{cases}$$

When SA succeeds (§3.2), this is known to be an upper bound for the differentiation index ν. In our experience it usually equals it: that is, SA finds the smallest possible set of differentiations.

As a reminder of notation for differentiated functions, suppose $x, y, \ldots$ are variables that depend on t, and f is a function of them and possibly of their first or higher t-derivatives $x', y', \ldots$. Then f' denotes $\mathrm{d}f/\mathrm{d}t$ treating the variables in f as functions of t. In general, it is a function of the variables in f, and also of derivatives to one order higher than those in f. For instance, if $f=x\sin(x'y)$—a function $f(x,x',y)$—then

$$f'=\frac{\partial f}{\partial x}x'+\frac{\partial f}{\partial x'}x''+\frac{\partial f}{\partial y}y'=\sin(x'y)x'+xy\cos(x'y)x''+xx'\cos(x'y)y',$$

a function $f'(x,x',x'',y,y')$. Higher derivatives $f''=f^{(2)}, f'''=f^{(3)},\ldots$ are defined inductively.

The staged solution process is defined for a DAE (1) by

Starting with $k = -\max d_j$, in order of increasing stage number k,

$$\text{solve the equations} \quad f_i^{(k+c_i)} = 0 \quad \text{for those } i \text{ such that } k + c_i \geq 0 \tag{5}$$

$$\text{for the unknowns} \quad x_j^{(k+d_j)} \quad \text{for those } j \text{ such that } k + d_j \geq 0. \tag{6}$$

This deceptively simple rule needs explanation. Define m_k and n_k to be the number of non-negative $k + c_i$, and $k + d_j$, respectively, so that (5, 6) specify m_k equations in n_k unknowns.

For $k < k_c := -\max_i c_i$, there are no equations to solve ($m_k = 0$), and for $k < k_d := -\max_j d_j$ there are no unknowns to solve for ($n_k = 0$). Since all finite σ_{ij} are ≥ 0, it follows from (2) that $k_d \leq k_c \leq 0$ (and $k_d < 0$, except in the special case that the DAE is a purely algebraic system). Thus the solution process starts at stage $k = k_d$.

It is immediate from their definition that the m_k and n_k increase with k, with $m_k \leq n_k$ for all k, and $m_k = n_k = n$ when $k \geq 0$.

The variables and derivatives found at a given stage occur (usually) in the equations at all later stages. Thus, globally over all the stages, the equations have a block-triangular structure that is solved by block forward substitution: known values from previous stages are substituted into the m_k equations (5) of the current stage, leaving just the n_k items (6) to solve for.

The block-triangular structure of the process for the pendulum DAE (4) is illustrated in Table I.

Table I. Block-triangular structure of the general solution scheme, shown for the Pendulum DAE. • marks values to be solved for at current stage; × marks values known from previous stages. Stages $-2, -1$ are under-determined.

Stage k		x, y	x', y'	x'', y'', λ	x''', y''', λ'	$\ldots$
-2	$C = 0$	•				
-1	$C' = 0$	×	•			
0	$A, B, C'' = 0$	×	×	•		
1	$A', B', C''' = 0$	×	×	×	•	
$\vdots$	$\vdots$					$\ddots$

3. CONSISTENT POINTS AND INITIAL VALUES

First, we introduce some convenient notation (§3.1). Then we discuss the SA view of consistent points and constraints (§3.2) and the desirability of minimizing the number of initial values required by an IVP (§3.3), a problem addressed by the block-triangular form (BTF) analysis introduced in the next section.

3.1. Notation for vectors of derivatives

Henceforth, for brevity, we write "derivatives of x_j" instead of "x_j and its derivatives"—derivatives $v^{(l)}$ of a variable v include v itself as the case $l = 0$. Let $\mathcal{J}$ be the set of index-pairs labeling arbitrary derivatives of the x_j, or of the f_i:

$$\mathcal{J} = \{\, (j, l) \mid 1 \leq j \leq n;\ l \in \mathbb{N} = \{0, 1, 2, \ldots\} \,\}, \tag{7}$$

A vector of specified derivatives of state variables of (1) can be denoted as $\mathbf{x}_J$ for some finite subset J of $\mathcal{J}$, meaning the vector of all derivatives $x_j^{(l)}$ for $(j,l) \in J$; it may be numeric, or symbolic, or a vector function $\mathbf{x}_J(t)$, depending on context.

For a DAE with individually named variables, such as the pendulum (4), we use a natural notation for vectors $\mathbf{x}$ of derivatives: for instance we may consider the vector

$$\mathbf{x} = (x, x', y, y').$$

This converts to the general notation thus: since x, y are x_1, x_2, we have

$$\mathbf{x} = \big(x_1^{(0)}, x_1^{(1)}, x_2^{(0)}, x_2^{(1)}\big),$$

so

$$\mathbf{x} = \mathbf{x}_J, \;\; J = \{(1,0),(1,1),(2,0),(2,1)\} \subset \mathcal{J}.$$

Similarly, the vector of functions

$$(A, B, C'') = (f_1^{(0)}, f_2^{(0)}, f_3^{(2)}) = \mathbf{f}_I, \text{ where } I = \{(1,0),(2,0),(3,2)\} \subset \mathcal{J}.$$

3.2. The SA view of consistent points

We say, see [Nedialkov and Pryce 2005, §4.4], that solutions of the DAE *live in* $\mathbf{x}_J$ *space*, if there is at most one solution having given values of the derivatives specified by $\mathbf{x}_J$ at $t = 0$—or at a given t, if the DAE is non-autonomous, i.e., if t occurs explicitly in it. We assume that the same J will do for all points along a solution path and for all solution paths (however many DAEs have "switching" behavior that violates this). Let us call an $\mathbf{x}_J$ space a *home*, if the DAE lives in it. A home is not unique: any "larger" space (an $\mathbf{x}_{J'}$ space with $J' \supset J$) has the same property.

Finding a home space amounts to finding what derivatives $x_j^{(l)}$ must be given IVs to define a unique solution, as argued in §3.3. So it is an important user-interface issue.

A point of a home space through which a solution of the DAE exists is called a *consistent point*. The set $\mathcal{M}$ of all consistent points is the *consistent manifold* for this space. Equations of which $\mathcal{M}$ is the zero-set are *constraints* of the DAE for this space.

Crucial for our solution method is the DAE's $n \times n$ *System Jacobian* matrix $\mathbf{J}$, where

$$\mathbf{J}_{ij} \;=\; \frac{\partial f_i}{\partial x_j^{(d_j - c_i)}} \;=\; \begin{cases} \dfrac{\partial f_i}{\partial x_j^{(\sigma_{ij})}} & \text{if } d_j - c_i = \sigma_{ij}, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

If there exists a consistent point at which $\mathbf{J}$ is nonsingular, then there is a solution of the DAE through it, at least locally, and we say the SA *succeeds*; otherwise it *fails*. In our experience, SA succeeds in most practical applications, but not all.

The problem of checking for success numerically is similar to that of finding a consistent point to start integrating an IVP, with the difference that there are no user-specified IVs to keep "close" to: any point will do that satisfies the consistency equations and has $\mathbf{J}$ nonsingular. Currently DAESA does not do a numerical success check. We aim to offer this in a future version.

When SA succeeds (and along the thus defined solution path), the equations (5, 6) form, for each $k > 0$, a square nonsingular linear system whose matrix is $\mathbf{J}$, which therefore has a unique solution. This also holds for $k = 0$ in the common case that the DAE is *quasilinear*, that is, the $x_j^{(d_j)}$ for $j = 1, \ldots, n$, occur linearly (or are absent) in f_i for $i = 1, \ldots, n$.

That is, once one has solved all stages $k \le 0$ ($k < 0$ if quasilinear), the solution is uniquely determined. Equivalently, one does not need to look for a home space larger than $x_{J_{\le 0}}$, or $x_{J_{<0}}$ if quasilinear, where the index sets

$$\begin{aligned} J_{\le 0} &= \{\, (j,l) \mid j = 1,\dots,n;\ l = 0,\dots,d_j \,\} \quad \text{and} \\ J_{<0} &= \{\, (j,l) \mid j = 1,\dots,n;\ l = 0,\dots,d_j - 1 \,\} \end{aligned} \tag{8}$$

list the $x_j^{(l)}$ solved for in stages $k \le 0$, and $k < 0$, respectively. Further, the constraints that define $\mathcal{M}$ are the set of equations $f_{I_{\le 0}} = 0$, when solutions live in $x_{J_{\le 0}}$ space, or $f_{I_{<0}} = 0$, when solutions live in $x_{J_{<0}}$ space, where

$$\begin{aligned} I_{\le 0} &= \{\, (i,l) \mid i = 1,\dots,n;\ l = 0,\dots,c_i \,\} \quad \text{and} \\ I_{<0} &= \{\, (i,l) \mid i = 1,\dots,n;\ l = 0,\dots,c_i - 1 \,\} \end{aligned} \tag{9}$$

list the $f_i^{(l)}$ solved for in stages $k \le 0$, and $k < 0$, respectively.

Example 3.1. For the pendulum DAE (4) the Jacobian is

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

where a blank means zero. We see the method succeeds, because $\det \mathbf{J} = -2(x^2 + y^2) = -2L^2 \neq 0$ at a consistent point. The DAE is quasilinear: once stage $k = -2$'s values x, y and stage $k = -1$'s values x', y' are known at a given t, a nonsingular linear system determines stage $k = 0$'s values x'', y'' and λ.

Hence, solutions live in 4-dimensional $(x, x'; y, y')$ space, and in this space the consistent manifold $\mathcal{M}$ is the 2-dimensional solution set defined by constraints

$$C = x^2 + y^2 - L^2 = 0 \quad \text{and} \quad C' = 2xx' + 2yy' = 0. \tag{10}$$

In the general notation, $\mathcal{M}$ is defined by $f_{I_{<0}} = 0$ in $x_{J_{<0}}$ space, where

$$J_{<0} = \{(1,0),(1,1);(2,0),(2,1)\} \quad \text{and} \quad I_{<0} = \{(3,0),(3,1)\}.$$

Constraints that are some f_i differentiated at least once are often called "hidden constraints". Here, C' is a "hidden" and C an "explicit" constraint.

As (8, 9) illustrate, home spaces of interest to us here and in Section 4 have the form "all x_j-derivatives up to some j-dependent order", and the corresponding constraints have the form "all f_i-derivatives $= 0$ up to some i-dependent order". Hence we define, for a non-negative integer vector $\delta = (\delta_1, \dots, \delta_n)$, the notation

$$\mathcal{J}(\delta) = \{\, (j,l) \mid j = 1,\dots,n;\ 0 \le l < \delta_j \,\} \tag{11}$$

a subset of $\mathcal{J}$ in (7).

In this notation, (8, 9) say that a home space is $\mathbf{x}_{\mathcal{J}(d)}$ with constraints $\mathbf{f}_{\mathcal{J}(c)} = 0$, if the DAE is quasilinear, and $\mathbf{x}_{\mathcal{J}(d+1)}$ with constraints $\mathbf{f}_{\mathcal{J}(c+1)} = 0$ otherwise. Here c and d are the offset vectors, and $c + 1$ denotes $(c_1 + 1, \dots, c_n + 1)$, and $d + 1$ similarly.

3.3. Minimizing the set of initial values

How should a user specify a numerical IVP? Ideally, by giving an exact consistent point: numerical values for some collection of derivatives, such that there is one and only one solution taking these values at the initial t. However, it is notoriously hard to make even a good guess at consistent values, especially of derivatives, so one would like an IVP code to minimize the number of IVs[2]: to ask the user for an $\mathbf{x}_J^*$ where $|J|$, the number of elements in J, is a minimum given that solutions live in $\mathbf{x}_J$ space.

Typically a code then finds a nearby $\mathbf{x}_J$ that is consistent (within a tolerance). E.g., for the pendulum, given a numerical $\mathbf{x}_J^* = (x^*, y^*, (x')^*, (y')^*)$ it might seek $\mathbf{x}_J = (x, y, x', y')$ that minimizes $\|\mathbf{x}_J - \mathbf{x}_J^*\|$ in some norm subject to the constraints (10) that define $\mathcal{M}$.

Common sense, and the stage-wise form of the solution process (5, 6), suggest the IVs should not include a derivative of some variable, if there is a lower derivative of the same variable that is not included. E.g., for the pendulum, maybe values for x, x'', y', λ specify a unique solution, but these are not sensible IVs to ask of a user.

Hence, we restrict attention to IV data of the form $\mathbf{x}_{\mathcal{J}(\delta)}$ as in (11). Since $|\mathcal{J}(\delta)| = \sum_j \delta_j$, we pose the problem thus:

$$\text{Seek a home space } \mathbf{x}_{\mathcal{J}(\delta)} \text{ with minimum } \textstyle\sum_j \delta_j.$$

As seen above, we can take δ to be d in the quasilinear case, and $d+1$ otherwise. However, for most DAEs in applications, this is far from minimizing $\sum_j \delta_j$. These *global offsets* d_j should be replaced by the *local offsets* $\widehat{d}_j$ derived from BTF analysis as discussed in Section 4.

Remark 3.2. Most current DAE codes do not have this ability to be selective about the IVs needed for a particular problem. E.g., the code DASSL requires "flat" data comprising all variables $(x_1, \ldots, x_n)$ and all first derivatives $(x_1', \ldots, x_n')$. This corresponds to $\delta = (2, 2, \ldots, 2)$ in the $\mathcal{J}(\delta)$ notation.

Remark 3.3. *Using the equations to reduce the IVs needed.* At stage k there are n_k unknowns (derivatives) to find, which must satisfy $m_k \le n_k$ equations. Often $m_k = 0$, in which case the unknowns are genuinely arbitrary initial values, as for an ODE. However when $m_k > 0$, there is the possibility of specifying $n_k - m_k$ values—and/or constraints of another kind—sufficient to determine all n_k unknowns uniquely. E.g., suppose the pendulum DAE has $L = 10$. At stage -2 with $m_{-2} = 1$ equation $x^2 + y^2 - 10^2 = 0$ to solve for $n_{-2} = 2$ unknowns x, y, one could specify that $x = 6$ is fixed, plus the constraint $y < 0$. This suffices to fix $y = -8$ uniquely. One can do similarly at stage -1. In general, $n_k - m_k$ is "the number of DOF introduced at stage k", and its sum over $k < 0$ is the total DOF of the system.

If done for all stages < 0 and based on BTF analysis, this approach has the merit of requiring exactly as many IVs as there are DOF, but software must take care to avoid problems of conditioning (in the above example, y is ill-determined when x is just under 10), existence (there is no y when $x > 10$) and uniqueness (one must know what constraints like "$y < 0$" to impose). Therefore we regard this idea as an issue of user interface, not of the theory presented here, and do not pursue it further.

4. BLOCK-TRIANGULAR FORM WITHIN THE DAE

We illustrate how "superfluous" IVs are suggested by the original Σ-method (§4.1), and show how block-triangular form (BTF), based on various sparsity patterns derived

[2]IV means both *known* initial values and also *trial* values, guesses at the solution of an equation system.

from the DAE, can reduce the number of IVs (§4.2, §4.3). We discuss irreducible BTF based on the Jacobian sparsity and the associated *local offsets* (§4.4, §4.5). Its synergy with quasilinearity analysis often reduces the number of IVs still further (§4.6).

4.1. Illustrating the initial value anomaly

The equations (5, 6) can do without initial values when k is large enough that they become square linear systems. However, this can happen for *parts* of the DAE before one reaches stage $k = 0$, as illustrated by the system 2PENDA, comprising two simple pendula with a coupling term:

$$\begin{aligned} &\text{2PENDA} \\ 0 = A &= x'' + x\lambda, \\ 0 = B &= y'' + y\lambda - G, \\ 0 = C &= x^2 + y^2 - L^2, \\ 0 = D &= u'' + u\mu, \\ 0 = E &= v'' + v\mu - G, \\ 0 = F &= u^2 + v^2 - (L + cx')^2, \end{aligned} \tag{12}$$

where G, L, c are constants. Its signature matrix and Jacobian are

$$\Sigma = \begin{array}{c|cccccc|c} & x & y & \lambda & u & v & \mu & c_i \\ A & 2 & & 0^\bullet & & & & 1 \\ B & & 2^\bullet & 0 & & & & 1 \\ C & 0^\bullet & 0 & & & & & 3 \\ \hline D & & & & 2 & & 0^\bullet & 0 \\ E & & & & & 2^\bullet & 0 & 0 \\ F & 1 & & & 0^\bullet & 0 & & 2 \\ \hline d_j & 3 & 3 & 1 & 2 & 2 & 0 & \end{array}, \qquad \mathbf{J} = \begin{array}{c|ccc|ccc} & x & y & \lambda & u & v & \mu \\ A & 1 & & x & & & \\ B & & 1 & y & & & \\ C & 2x & 2y & & & & \\ \hline D & & & & 1 & & u \\ E & & & & & 1 & v \\ F & \xi & & & 2u & 2v & \end{array}.$$

where $\xi = -2c(L + cx')$

(A blank in Σ means $-\infty$, and in $\mathbf{J}$ means zero.)

One can see just from Σ (or from $\mathbf{J}$), without studying the equations themselves, that the system splits into parts, i.e., subsystems: part 1 has equations A, B, C for variables x, y, λ; part 2 has equations D, E, F for variables u, v, μ. Part 1 influences part 2 by the F, x entry in the lower left block of Σ, but is uninfluenced by it since the top right block is blank, thus giving a block-triangular form (BTF).

This coupling leaves the offsets of part 2 unchanged, but increases by one those of part 1—from $c_i = 0, 0, 2,\ d_j = 2, 2, 0$ to $c_i = 1, 1, 3,\ d_j = 3, 3, 1$. This seems to change the initial values part 1 requires—paradoxical, since part 1 is the "uninfluenced" one. Namely, the combined DAE is quasilinear, so according to (8), IVs are needed for

$$x_{J_{<0}} = (x, x', x''; \ y, y', y''; \ \lambda; u, u'; \ v, v').$$

So part 1 now seems to need IVs for x'', y'' and λ, which as a stand-alone system it did not. Of course this is false.

When one considers the offsets as defining a scheme for computing successive derivatives (equivalently, Taylor coefficients), the reason for the raised offsets of pendulum 1 is clear. In the absence of coupling, the natural scheme is: at stage $k = -2$, find x, y and u, v; at $k = -1$, find x', y' and u', v'; at $k = 0$, find x'', y'', λ and u'', v'', μ; and so on. However, u, v must satisfy $F = 0$. With the coupling, F involves x', which in the above

scheme has not been found yet. Similarly, u', v' must satisfy $F' = 0$, which involves x'', which has not been found yet, and so on. This is cured by shifting pendulum 1 back one stage, so that the scheme becomes:

k	find	*then* find
-3	x, y	
-2	x', y'	u, v
-1	x'', y'', λ	u', v'
0	x''', y''', λ'	u'', v'', μ

and so on. Hence, provided pendulum 1's equations are solved before those of pendulum 2 at each k-stage,

— pendulums 1 and 2 can be solved as separate size 3 systems;
— but, each derivative of x is available just when needed by pendulum 2.

Because of the shift, pendulum 1 behaves as if the overall stages $k = -3, -2, -1, \ldots$ are its *local stages* $\widehat{k} = k + 1 = -2, -1, 0, \ldots$ associated with *local offsets* $\widehat{c}_i = 0, 0, 2$ and $\widehat{d}_j = 2, 2, 0$, which come from analyzing it as a stand-alone system. This shows that the relation between the "minimal initial values" problem and the sequencing of a solution scheme involves the DAE's block-triangular structure.

4.2. Sparsity patterns and BTF

A BTF of (1) is obtained from a *sparsity pattern*, which is some subset A of $\{1, \ldots, n\}^2$, the $n \times n$ matrix positions (i, j) for i and j from 1 to n. When appropriate, we identify A with its $n \times n$ *incidence matrix*

$$(a_{ij}) \text{ where } a_{ij} = 1 \text{ if } (i, j) \in A, \text{ 0 otherwise.}$$

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

$$S = \{\, (i, j) \mid \sigma_{ij} > -\infty \,\} \qquad \text{(the } \textit{sparsity pattern of } \Sigma\text{).} \tag{13}$$

However, a more informative BTF comes from the sparsity pattern of the Jacobian $\mathbf{J}$ (but see the caveats in §4.3). It depends, as does $\mathbf{J}$, on the (valid) offsets c, d used:

$$S_0 = S_0(c, d) = \{\, (i, j) \mid d_j - c_i = \sigma_{ij} \,\} \qquad \text{(the } \textit{sparsity pattern of } \mathbf{J}\text{).} \tag{14}$$

A less obvious, but useful, set [Nedialkov and Pryce 2005] is

$$S_{\text{ess}} = \text{the union of all HVTs of } \Sigma. \qquad \text{(the } \textit{essential sparsity pattern}\text{).}$$

(S_{ess} was called S_{00} in [Pryce 2001].) Both S and S_{ess} depend only on the DAE, while to compute an S_0 one needs to find c, d first. Since $d_j - c_i = \sigma_{ij}$ holds on each HVT by (3), and implies $\sigma_{ij} > -\infty$, we have

$$S_{\text{ess}} \subseteq S_0(c, d) \subseteq S \qquad \text{for any } c, d. \tag{15}$$

Experience suggests that in applications, a BTF based on S_0 is usually significantly finer than one based on S. We refer to the former as a *fine* BTF, and we refer to the latter as *coarse* BTF.

A permuted block form of the DAE is created by forming permutations $(\widetilde{f}_1, \ldots, \widetilde{f}_n)$ and $(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ of the vectors of equations and variables, and then writing these in block form $(F_1, \ldots, F_p)$ and $(X_1, \ldots, X_p)$, where subvectors F_l and X_l have the same number $N_l > 0$ of elements for each l, and $N_1 + \cdots + N_p = n$. Then Σ, $\mathbf{J}$ and a sparsity pattern A are permuted and put in block form correspondingly. We ignore the permutations, i.e., suppose them already done (in examples we make them obvious by labeling

rows and columns with the equation and variable names), and write in $p \times p$ block form:

$$\Sigma = \begin{bmatrix} \Sigma_{11}, & \cdots, & \Sigma_{1p} \\ \vdots & & \vdots \\ \Sigma_{p1}, & \cdots, & \Sigma_{pp} \end{bmatrix}, \quad \mathbf{J} = \begin{bmatrix} \mathbf{J}_{11}, & \cdots, & \mathbf{J}_{1p} \\ \vdots & & \vdots \\ \mathbf{J}_{p1}, & \cdots, & \mathbf{J}_{pp} \end{bmatrix}, \quad A = \begin{bmatrix} A_{11}, & \cdots, & A_{1p} \\ \vdots & & \vdots \\ A_{p1}, & \cdots, & A_{pp} \end{bmatrix}, \tag{16}$$

where the (l, m) block is of size N_l by N_m.

Let $\mathrm{blockOf}(i)$ denote the block that index i belongs to, that is

$$\mathrm{blockOf}(i) = l \quad \text{if} \quad \textstyle\sum_{m=1}^{l-1} N_m < i \leq \sum_{m=1}^{l} N_m, \tag{17}$$

where $1 \leq i \leq n$, $1 \leq l \leq p$.

DAESA uses *upper* BTF, so Σ is in BTF if the below-diagonal blocks (where $l > m$) are all $-\infty$; while $\mathbf{J}$, and A viewed as an incidence matrix, are in BTF if the below-diagonal blocks are zero. When A is viewed as a set of (i, j)'s, it is the disjoint union of its subsets

$$A_{lm} = \{\, (i, j) \in A \mid \mathrm{blockOf}(i) = l \text{ and } \mathrm{blockOf}(j) = m \,\},$$

and is in BTF if the below-diagonal blocks are empty.

Basing BTF on a sparsity pattern S_0 of (14) will of course put the corresponding $\mathbf{J}$ into BTF, but may not do so for Σ, since there may be elements of the (permuted) pattern S of (13) below the block diagonal, if S_0 is a proper subset of it.

The following facts about transversals are relevant. Fuller proofs of these and other results in Section 4 are in the future article [Tan et al. 2012a].

LEMMA 4.1.

(i) Any transversal T of (i.e., contained in) a sparsity pattern A is contained in the union of the diagonal blocks of any BTF of A.

(ii) Any transversal T of a Jacobian sparsity pattern $S_0(c, d)$, see (14), is a HVT of Σ.

The first of these is well known; for the second, we have

$$\begin{aligned} \mathrm{Val}(\Sigma) &= \textstyle\sum_{j=1}^{n} d_j - \sum_{i=1}^{n} c_i && \text{since } c, d \text{ are valid offsets} \\ &= \textstyle\sum_{(i,j)\in T} (d_j - c_i) && \text{since } T \text{ is a transversal} \\ &= \textstyle\sum_{(i,j)\in T} \sigma_{ij} && \text{since } T \subseteq S_0 \\ &= \mathrm{Val}(T) && \text{by definition,} \end{aligned}$$

proving that T is a HVT.

4.3. Caveats

Some comments are needed about sparsity analysis. First, if the DAE is structurally well-posed (§2.2), an analysis based on S_0 can always be carried out. However, it is only meaningful if the method is known to succeed: that is, if a consistent point where $\mathbf{J}$ is nonsingular has been found. Second, S_0 is the set of positions where $\mathbf{J}_{ij}$ is nonzero (a) structurally and (b) as far as DAESA can tell from the equations as written.

As regards (a), $\mathbf{J}$ varies along the solution, and a generically nonzero $\mathbf{J}_{ij}$ may vanish at certain t values. For (b), algebraic cancellation may make some $\mathbf{J}_{ij}$ identically zero for reasons that DAESA does not detect. For instance, suppose equation f_1 is $(tx_1)' - tx_1' + x_2 = 0$, which simplifies to $x_1 + x_2 = 0$. DAESA applies quasilinearity analysis based on an operator-overloading computation and does not do symbolic simplification, so it finds $\sigma_{11} = 1$ instead of the true $\sigma_{11} = 0$. Such overestimation of elements of Σ looks dangerous, but—provided a nonsingular $\mathbf{J}$ is in due course found—can do no worse than cause some zero entries of $\mathbf{J}$ to be treated as nonzero, and make numerical solution slightly less efficient than it could be: see [Nedialkov and Pryce 2005, §5].

4.4. Irreducibility

An $n \times n$ sparsity pattern A (or matrix) is *irreducible*, if it cannot be permuted to a non-trivial BTF (one with $p > 1$); otherwise it is reducible.

Classic results of graph theory, see e.g. [Pothen and Fan 1990; Coleman et al. 1986], are the following. The *bipartite graph* of A is the undirected graph whose $2n$ vertices are the n rows and n columns, and which has an edge between row i and column j whenever $(i, j) \in A$.

THEOREM 4.2.

(a) The following are equivalent:
- *(i) A is structurally nonsingular (contains at least one transversal).*
- *(ii) A has the Hall Property: for $r = 1, \ldots, n$, any set of r columns of A contains elements of at least r rows.*

(b) The following are equivalent:
- *(i) A is irreducible.*
- *(ii) A has the Strong Hall Property: for $r = 1, \ldots, n-1$, any set of r columns of A contains elements of at least $r+1$ rows.*
- *(iii) Every element of A is on a transversal, and the bipartite graph of A is connected.*

Clearly, if any diagonal block A_{ll} of a BTF of A is reducible (as a sparsity pattern in its own right), this lets one refine the whole BTF of A, splitting the lth block into two or more smaller blocks. Repeating this as needed, A has a BTF for which each diagonal block is irreducible, which we call an *irreducible BTF* of A. This is unique up to possible reordering of the blocks, [Duff et al. 1986].

The $\subseteq$ part of the following result comes from Lemma 4.1(i,ii) and the definition of S_{ess}. The $\supseteq$ part comes from the fact, which uses Theorem 4.2 (b)(iii), that each element of each diagonal block (S_{ll}) is on a transversal.

THEOREM 4.3. *For a given signature matrix Σ, let (S_{lm}) be an irreducible BTF of a Jacobian sparsity pattern $S_0 = S_0(c, d)$; that is,*

$$S_{lm} = \{\, (i,j) \in S_0 \mid \text{blockOf}(i) = l \text{ and } \text{blockOf}(j) = m \,\}.$$

Then the essential sparsity pattern S_{ess} is exactly the union of the diagonal blocks,

$$S_{ess} = S_{11} \cup \ldots \cup S_{pp}.$$

It follows that, up to possible reordering, all Jacobian sparsity patterns $S_0(c, d)$ of Σ have the same block sizes $N_1, \ldots, N_p$, and indeed identical diagonal blocks, in their irreducible BTFs. Depending on the DAE, some ordering of blocks may be arbitrary; some may be dictated by a specific choice of offsets c and d; some may be inherent in the DAE. This is studied in [Tan et al. 2012a].

4.5. BTF, local offsets, and solution scheme

For any BTF (16) of a DAE, *local offsets* of part l ($l = 1, \ldots, p$) are offsets obtained by structural analysis of Σ_{ll}, i.e., by treating part l as a free-standing DAE, with known driving terms coming from any coupling to parts $l{+}1, \ldots, p$. Since we are ignoring the permutations, we write a set of global offsets—*arbitrary* valid offsets of Σ as a whole—as $c_1, \ldots, c_n$ and $d_1, \ldots, d_n$ in the order of the permuted matrices in (16); and the *canonical* local offsets as $\widehat{c}_1, \ldots, \widehat{c}_n$ and $\widehat{d}_1, \ldots, \widehat{d}_n$ in the same way. (E.g., $d_1, \ldots, d_{N_1}$ and $\widehat{d}_1, \ldots, \widehat{d}_{N_1}$ are the global and local offsets of the variables of part 1 of the DAE, and so on.)

From [Tan et al. 2012a] we have the following, whose proof relies on Theorem 4.2 (b)(iii): an irreducible pattern has a connected bipartite graph.

THEOREM 4.4. *Consider the irreducible BTF in of the Jacobian sparsity pattern $S_0(c,d)$ for any valid (global) offsets c, d. Then the difference between global and local offsets is constant on each block. That is, there are non-negative integers $K_1, \ldots, K_p$ such that*

$$c_j - \widehat{c}_j = d_j - \widehat{d}_j = K_l \quad \textit{where } l = \mathrm{blockOf}(j), \quad \textit{see (17).}$$

Consider (5, 6) as a Taylor series generation scheme. Using the fine BTF, one can regard part l for $l = 1, \ldots, p$ as a separate DAE, possibly receiving driving terms from parts $l+1, \ldots, p$, the process being interleaved so that at each k-stage, needed Taylor coefficients are available for substitution into the equations just when they are needed.

Theorem 4.4 shows that this works, because each part behaves as if its private solution scheme, including the negative stages, is the same as if it were free-standing using its local offsets, but shifted K_l stages earlier. In particular, each part only requires the IVs that are specified by its *local* offsets. In future article [Tan et al. 2012a] we prove, with the notation and assumptions of §3.3:

THEOREM 4.5. *The minimal vector $\mathbf{x}_{\mathcal{J}(\delta)}$ of initial values of the DAE is based on the canonical local offsets, namely the vector δ is given by*

$$\delta_j = \begin{cases} \widehat{d}_j & \textit{if part } \mathrm{blockOf}(j) \textit{ of the DAE is quasilinear,} \\ \widehat{d}_j + 1 & \textit{otherwise.} \end{cases} \tag{18}$$

The number K_l can be viewed as a "lead time" given to each k-stage of part l of the DAE to ensure successful interleaving.

A *detailed solution scheme*, which is the refinement of (5, 6) based on the irreducible blocks of S_0 together with quasilinearity analysis, is displayed by DAESA's `printSolScheme` function. It gives valuable insight into the DAE and suggests ways to improve computational efficiency.

4.6. Quasilinearity and local quasilinearity

The quasilinearity referred to in (18) is of course local, meaning that part l is quasilinear regarded as a free-standing DAE—that is, the $\widehat{d}_j$th derivative of x_j, for all j belonging to block l, occurs linearly (or is absent) in f_i for all i belonging to block l.

Local quasilinearity occurs frequently, which is one reason why linearity analysis and BTF analysis together are more effective than either separately at reducing the number of IVs required. We call the DAE as a whole *locally quasilinear*, if each block in the irreducible BTF based on S_0 is locally quasilinear. Since these blocks are the same as those of S_{ess} by Theorem 4.3, this is a well-defined property independent of the (global) offsets c, d used to define S_0.

[Tan et al. 2012b] describes the algorithms used in DAETS and DAESA to determine quasilinearity.

5. SOME EXAMPLES

We give some examples to illustrate these ideas. The facts reported were confirmed using DAESA.

For a notation independent of permutations, matrix entries and offsets in these examples are labeled by their equation and/or variable instead of numerically, e.g., $c_A, c_B, \ldots, d_x, d_y, \ldots, \sigma_{A,x}, \ldots$ instead of $c_1, c_2, \ldots, d_1, d_2, \ldots, \sigma_{11}, \ldots$.

Example 5.1. For the 2PENDA system (12), the sparsity patterns S and S_0 give the same block structure: we show its signature matrix and Jacobian, permuted to upper BTF, annotated by both global and local offsets, and with the HVT marked with $\bullet$ placed on the main diagonal.

2PENDA

$$\Sigma = \begin{array}{c|cccccc|cc} & u & v & \mu & x & y & \lambda & c_i & \widehat{c}_i \\ F & 0^\bullet & 0 & & 1 & & & 2 & 2 \\ E & & 2^\bullet & 0 & & & & 0 & 0 \\ D & 2 & & 0^\bullet & & & & 0 & 0 \\ \hline C & & & & 0^\bullet & 0 & & 3 & 2 \\ B & & & & & 2^\bullet & 0 & 1 & 0 \\ A & & & & 2 & & 0^\bullet & 1 & 0 \\ \hline d_j & 2 & 2 & 0 & 3 & 3 & 1 & & \\ \widehat{d}_j & 2 & 2 & 0 & 2 & 2 & 0 & & \end{array}, \quad \mathbf{J} = \begin{array}{c|cccccc} & u & v & \mu & x & y & \lambda \\ F & 2u & 2v & & \xi & & \\ E & & 1 & v & & & \\ D & 1 & & u & & & \\ \hline C & & & & 2x & 2y & \\ B & & & & & 1 & y \\ A & & & & 1 & & x \end{array}.$$

where $\xi = -2c(L+cx')$

(19)

The DAE is locally quasilinear. Although the global offsets d_j indicate x'', y'' and λ need to be given as IVs, the local offsets show this is not so. Sufficient IVs are $x, y; u, v; x', y'; u', v'$.

Example 5.2. In 2PENDA of (12) and (19), Σ and $\mathbf{J}$ have the same sparsity pattern, i.e. $S = S_0$. A small change shows the BTF based on the sparsity of $\mathbf{J}$ can be finer than that based on the sparsity of Σ. Namely change equation A to, say,

$$A: \quad x'' + x\lambda + cu = 0. \tag{20}$$

This produces 2PENDB, with the signature matrix entry $\sigma_{A,u}$ (in the original form (12) it is σ_{14}) changed from $-\infty$ to 0. Permuting as in (19) gives:

2PENDB

$$\Sigma = \begin{array}{c|cccccc|cc} & u & v & \mu & x & y & \lambda & c_i & \widehat{c}_i \\ F & 0^\bullet & 0 & & 1 & & & 2 & 2 \\ E & & 2^\bullet & 0 & & & & 0 & 0 \\ D & 2 & & 0^\bullet & & & & 0 & 0 \\ \hline C & & & & 0^\bullet & 0 & & 3 & 2 \\ B & & & & & 2^\bullet & 0 & 1 & 0 \\ A & 0 & & & 2 & & 0^\bullet & 1 & 0 \\ \hline d_j & 2 & 2 & 0 & 3 & 3 & 1 & & \\ \widehat{d}_j & 2 & 2 & 0 & 2 & 2 & 0 & & \end{array}, \quad \mathbf{J} = \begin{array}{c|cccccc} & u & v & \mu & x & y & \lambda \\ F & 2u & 2v & & \xi & & \\ E & & 1 & v & & & \\ D & 1 & & u & & & \\ \hline C & & & & 2x & 2y & \\ B & & & & & 1 & y \\ A & & & & 1 & & x \end{array}.$$

where $\xi = -2c(L+cx')$

This does not change the offsets; but there is now two-way coupling between the pendula, and the BTF based on S is one irreducible block. However, the new entry $\sigma_{A,u} = 0$ is strictly less than $d_u - c_A = 2 - 1 = 1$, so it does not give a new element of S_0 (a nonzero in $\mathbf{J}$). Thus the BTF based on S_0 is the same as with 2PENDA.

As a DAE, pendulum 1 still drives pendulum 2—the reverse effect is too weak to change the solution scheme's block structure. Therefore the local offsets are also unchanged, and $x, y; u, v; x', y'; u', v'$ still suffice as IVs.

Example 5.3. Now consider 2PENDC, in which the cu term in (20) is changed to cu'. This *still* does not change the global offsets; but now $\sigma_{A,u} = 1 = d_u - c_A$, so $\mathbf{J}$ has a (structural) nonzero here, namely $\mathbf{J}_{A,u} = c$, and the two-way coupling is stronger. Now $S_0 = S$, and the BTF based on S_0 is one irreducible block.

Thus the local offsets are now the same as the global ones; each solution stage (for $k \geq 0$) entails solving a 6×6 linear system instead of two 3×3 ones; and $x, y; u, v; x', y'; u', v'$ no longer suffice as IVs—one must provide x'', y'' and λ also.

Example 5.4. All of 2PENDA, 2PENDB and 2PENDC have 4 DOF and index 4; but if we go one further and form 2PENDD, in which the cu term in (20) becomes cu'', the nature of the system alters completely. It now has 5 DOF and index 3.

The unpermuted Σ, showing the (unique) HVT and the global offsets, is below. The dividing lines show the two nominal subsystems, but have no relation to the true BTF.

2PENDD

$$\Sigma = \begin{array}{c|cccccc|c} & x & y & \lambda & u & v & \mu & c_i \\ A & 2 & & 0 & 2^\bullet & & & 0 \\ B & & 2 & 0^\bullet & & & & 0 \\ C & 0 & 0^\bullet & & & & & 2 \\ \hline D & & & & 2 & & 0^\bullet & 0 \\ E & & & & & 2^\bullet & 0 & 0 \\ F & 1^\bullet & & & 0 & 0 & & 1 \\ \hline d_j & 2 & 2 & 0 & 2 & 2 & 0 & \end{array}$$

In 2PENDA to C, the (F, u) and (F, v) entries in Σ gave rise to nonzero Jacobian entries, but now we have $\sigma_{F,u} = 0 < 1 = d_u - c_F$ and $\sigma_{F,v} = 0 < 1 = d_v - c_F$. Thus the entries $\mathbf{J}_{F,u}$ and $\mathbf{J}_{F,v}$ are now structural zeros of $\mathbf{J}$.

As a result the fine BTF now has six blocks of size 1. The permuted signature matrix with global and local offsets, and corresponding Jacobian, are

2PENDD

$$\Sigma = \begin{array}{c|cccccc|cc} & v & \mu & u & \lambda & y & x & c_i & \widehat{c}_i \\ E & 2^\bullet & 0 & & & & & 0 & 0 \\ D & & 0^\bullet & 2 & & & & 0 & 0 \\ A & & & 2^\bullet & 0 & & 2 & 0 & 0 \\ B & & & & 0^\bullet & 2 & & 0 & 0 \\ C & & & & & 0^\bullet & 0 & 2 & 0 \\ F & 0 & & 0 & & & 1^\bullet & 1 & 0 \\ \hline d_j & 2 & 0 & 2 & 0 & 2 & 2 & & \\ \widehat{d}_j & 2 & 0 & 2 & 0 & 0 & 1 & & \end{array}, \qquad \mathbf{J} = \begin{array}{c|cccccc} & v & \mu & u & \lambda & y & x \\ E & 1 & v & & & & \\ D & & u & 1 & & & \\ A & & & c & x & & 1 \\ B & & & & y & 1 & \\ C & & & & & 2y & 2x \\ F & & & & & & \xi \end{array}.$$

where $\xi = -2c(L + cx')$

The IVs comprise $x, y; x', y'; u, v; u', v'$, of which each of y and x' is a guess at the solution of a single nonlinear equation, and the remaining five are arbitrary, unconstrained,

values. A solution process is as follows:

- k=-2, 4 blocks:
  — blocks (F,x), (A,u), (E,v): solve nothing, but give IVs for x,u,v
  — block (C,y): give a guess for y and solve (nonlinear) $0=C=x^2+y^2-L^2$ for y
- k=-1, 4 blocks:
  — block (F,x): use u,v, give a guess for x', solve (nonlinear) $0=F=u^2+v^2-(L+cx')^2$ for x'
  — block (C,y): use x,x',y, solve $0=C'=2xx'+2yy'$ for y'.
  — blocks (A,u), (E,v): solve nothing, but give initial values for u', v'
- k=0, 6 blocks:

block	use	solve	for
(F,x)	x',u,u',v,v'	$0=F'=\left(u^2+v^2-(L+cx')^2\right)'$	x''
(C,y)	x,x',x'',y,y'	$0=C''=\left(x^2+y^2-L^2\right)''$	y''
(B,λ)	y,y''	$0=B=y''+y\lambda-G$	λ
(A,u)	x,x'',λ	$0=A=x''+x\lambda+cu''$	u''
(D,μ)	u,u''	$0=D=u''+u\mu$	μ
(E,v)	v,μ	$0=E=v''+v\mu-G$	v''

- k>0. Solve in order $0=F^{(k+1)}$ for $x^{(k+2)}$, $0=C^{(k+2)}$ for $y^{(k+2)}$, $0=B^{(k)}$ for $\lambda^{(k)}$, $0=A^{(k)}$ for $u^{(k+2)}$, $0=D^{(k)}$ for $\mu^{(k)}$, and $0=E^{(k)}$ for $v^{(k+2)}$

Example 5.5. *Chemical Akzo Nobel problem*
This problem, from the ODE/DAE test set [Mazzia and Iavernaro 2003], is a DAE of index 1 that, with a small change of notation, can be written as:

$$\begin{aligned}
0=f_1 &= -y_1' \;-2r_1\; +r_2 \;\; -r_3 \;\; -r_4,\\
0=f_2 &= -y_2' \;-\tfrac{1}{2}r_1 \qquad\qquad -r_4 \;-\tfrac{1}{2}r_5 \;+F_{\text{in}},\\
0=f_3 &= -y_3' \;\; +r_1 \;-r_2 \;+r_3,\\
0=f_4 &= -y_4' \qquad -r_2 \;+r_3 \;-2r_4,\\
0=f_5 &= -y_5' \qquad +r_2 \;-r_3 \qquad\quad +r_5,\\
0=f_6 &= \qquad\quad K_s\, y_1\, y_4 - y_6\,,
\end{aligned} \tag{21}$$

where r_1 to r_5 and F_{in} are auxiliary variables given by

$$r_1 = c_1\, y_1^4\, y_2^{\frac{1}{2}},\ r_2 = c_2\, y_3\, y_4,\ r_3 = c_3\, y_1\, y_5,$$
$$r_4 = c_4\, y_1 y_4^2,\ r_5 = c_5\, y_6^2\, y_2^{\frac{1}{2}},\ F_{\text{in}} = c_6(c_7 - y_2),$$

and K_s and the c_i are positive constants.

The signature matrix is below left. According to SA based on its sparsity pattern, the problem is irreducible. Further, the r_i are nonlinear in the y_i, so it appears the solution process involves a fully nonlinear system of size 6.

However, SA gives a different picture. The only off-diagonal entries where $d_j - c_i = \sigma_{ij}$ are in the last column. Thus $\mathbf{J}$, below right, is much sparser than Σ and is already triangular without reordering.

$$\Sigma = \begin{array}{c|cccccc|c} & y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & c_i \\ \hline f_1 & 1^\bullet & 0 & 0 & 0 & 0 & & 0 \\ f_2 & 0 & 1^\bullet & & 0 & & 0 & 0 \\ f_3 & 0 & 0 & 1^\bullet & 0 & 0 & & 0 \\ f_4 & 0 & & 0 & 1^\bullet & 0 & & 0 \\ f_5 & 0 & 0 & 0 & 0 & 1^\bullet & 0 & 0 \\ f_6 & 0 & & & 0 & & 0^\bullet & 0 \\ \hline d_j & 1 & 1 & 1 & 1 & 1 & 0 & \end{array}, \qquad \mathbf{J} = \begin{array}{c|cccccc} & y_1 & y_2 & y_3 & y_4 & y_5 & y_6 \\ \hline f_1 & -1^\bullet & & & & & \\ f_2 & & -1^\bullet & & & & -c_5 y_6 y_2^{\frac{1}{2}} \\ f_3 & & & -1^\bullet & & & \\ f_4 & & & & -1^\bullet & & \\ f_5 & & & & & -1^\bullet & 2c_5 y_6 y_2^{\frac{1}{2}} \\ f_6 & & & & & & -1^\bullet \end{array} \tag{22}$$

Hence the fine BTF has six blocks of size 1. In fact each is quasilinear, so that the solution scheme only solves linear systems of size 1. Namely at stage $k = -1$, give arbitrary IVs for y_1, y_2, y_3, y_4, y_5. At stage $k = 0$, solve f_6 for y_6, then f_i for y_i', $i = 1, \ldots, 5$ (all linear). See also §5.3 in the companion paper.

Example 5.6. A more extreme example of the saving in IVs achieved by using local offsets is the multi-pendula "chain" in §6.1 of the companion paper. This is similar to the 2PENDA,B,C,D examples, but with p pendula coupled in such a way that each new one added increases the global offsets of the first, "independent" pendulum—as well as the index of the whole system—by 2. With the correct analysis, based on the local offsets, the whole system requires $4p$ initial values (an x, y, x' and y' for each pendulum) as one expects; but analysis based on the global offsets requires $O(p^2)$ IVs.

6. IMPLEMENTATION ISSUES

We discuss finding global and local offsets (§6.1), analyzing linearity (§6.2), and possibilities for parallelism (§6.3).

6.1. Efficiently finding the global and local offsets

As noted above, block-triangularization based on S_0 can only be done after the global offsets are computed. However, our goal is to handle large systems with n up to 10^5 or greater. Solving the linear assignment problem to find a HVT is a large part of the cost for such DAEs—the standard algorithms have $O(n^3)$ complexity for a dense matrix. Even though this is more like $O(n^2)$ for the sparse case, using an initial BTF usually pays off handsomely, because the HVT can be found for each diagonal block separately. (For instance, if a sparse system of size $n = Np$ has p blocks of size N, then solving separately costs $\approx p \cdot CN^2$ compared with $\approx C(Np)^2$ for solving all at once, saving a factor of $\approx p$.)

Hence DAESA uses two passes. The first finds a coarse BTF based on S. In the second pass, the LAP is solved, and resulting (canonical) offsets are found for each coarse block. This leads to a local BTF of fine blocks within each coarse block. The local offsets of each fine block are found here. (Since the off-block-diagonal elements are irrelevant here, the second pass, up to this point, could be done in parallel over coarse blocks.) Taking account of Σ entries outside the coarse diagonal blocks, one further pass over the fine blocks assembles the global offsets of the whole system from the local offsets. (This part is sequential over the fine blocks; finding the local offsets of each fine block can be done in this pass, depending on how much parallelism one uses.)

Our experience is that almost all DAEs of nontrivial size in applications have a useful fine block structure, and that the size of blocks varies widely.

For a large example, we constructed a DAE whose Σ has the sparsity pattern of the LHR71 matrix from the Florida collection [Davis and Hu 2011], with the finite σ_{ij}

being 0, 1 or 2 at random. The system is of size $n = 70304$; its block structure comprises 15774 blocks of size 1, four of size 8, 154 of size in the tens, 24 of size in the hundreds, and 14 of size in the thousands, the largest being two of size 4586.

6.2. Analyzing linearity

DAESA verifies whether any block of equations and variables is linear in the variables to be solved for, using operator overloading and a variant of the automatic differentiation method by which the signature matrix is computed.

The analysis is made easier by some simple facts. The first, which follows from (2, 5, 6), is that in any single, undifferentiated equation f_i, the unknowns to be solved for—however the system may have been decomposed into triangular form—are always a subset of the highest-order derivatives, that is of the $x_j^{(\sigma_{ij})}$, $j = 1, \ldots, n$. For instance if f_i involves x_2', x_4 and x_4'' then x_2' and x_4'' may be among the unknowns to be solved for, but x_4 cannot be.

The second fact is that in any function differentiated (w.r.t. t) one or more times, all the highest derivatives occur linearly. For instance if f_i is a function only of (x_2', x_4, x_4'') then f_i' is a function only of $(x_2', x_2'', x_4, x_4', x_4'', x_4''')$, namely

$$f_i' = \frac{\partial f_i}{\partial x_2'}(x_2', x_4, x_4'')\, x_2'' + \frac{\partial f_i}{\partial x_4}(x_2', x_4, x_4'')\, x_4' + \frac{\partial f_i}{\partial x_4''}(x_2', x_4, x_4'')\, x_4''',$$

which is linear in its highest derivatives x_2'' and x_4'''.

Hence each f_i needs exactly one linearity check, at the unique stage in the sequence (5, 6) where it occurs undifferentiated: namely for $k = -c_i$. It is checked against the set of unknowns in its equation-block, which depends on the chosen BTF. At earlier stages the equation does not appear; at later ones, it appears differentiated, so is necessarily linear.

6.3. Parallelizing the solution of IVPs

For large systems, parallel computation may be useful during the process of finding a block-triangular form, as noted above. It also seems a promising approach for numerical integration of IVPs, exploiting BTF.

One way is as follows. Some coarse BTF, based on S, is used to handle blocks of equations by separate threads on the machine, possibly on separate processors. Let the lth block of equations be $F^{(l)} = 0$, to solve for variables $X^{(l)}$, $l = 1, \ldots, q$. The off-diagonal block structure defines a directed acyclic graph: its vertices are the threads, and an edge to l from $l' > l$ means l uses l', i.e., $F^{(l)}$ contains variables from $X^{(l')}$. Each thread l is responsible for generating a sufficiently smooth piecewise polynomial representation $P^{(l)}(t)$ of its solution function $X^{(l)}(t)$, and as many derivatives as may be needed by other threads.

The threads have their own independent step size control, so that at any moment they have in general integrated up to different points $t = \tau^{(l)}$. Integration is advanced on demand. The process is kicked off by telling each *output* thread l (one which no other threads use) to integrate from $t = 0$ to some $t = T$. Each of these chooses a trial step, say to $t = h^{(l)}$. This generates a demand that all blocks, which this block uses, integrate at least as far over $[0, h^{(l)}]$ to provide their needed polynomial approximants on this interval. These demands cascade back so that each thread, independently, is made to integrate at least as far as the smallest of the $h^{(l)}$'s over all output threads.

Once each output thread l has achieved a successful step it repeats the process until it reaches $t = T$, thus forcing all other threads to integrate up to T. Such a scheme allows the integrators of different blocks considerable freedom. For instance one block

might be "difficult" requiring small steps, which would not need to impact on other blocks that might be able to take larger steps.

Such a scheme is generally not possible using a fine BTF based on S_0, because $l' > l$ may not imply block l' is completely independent of block l. Thus one cannot do several integration steps (or even one) of l' independently of l, by contrast with the coarse BTF case.

7. SUMMARY OF DAESA FEATURES, AND FUTURE WORK

DAESA's facilities, given in detail in [Nedialkov et al. 2012], are outlined here. The analysis is begun by passing MATLAB code of the DAE to the `daeSA` function, which returns the results of its analysis in a MATLAB structure. Other functions can then extract the information for further use in a program and/or display it, for example:

— Produce a graphical display of the signature matrix with global, and local if appropriate, offsets; either in its original form or permuted into the coarse BTF based on S, or the fine BTF based on S_0.
— Display the refined solution scheme based on the fine BTF, indicating which sets of equations are underdetermined, which are quasilinear, etc.
— List a minimal set of derivatives required as IVs, obtained from the solution scheme.

Taken together, these features go beyond the structural analysis provided by any other current DAE solver or simulation system.

There is currently one serious omission—a numerical check that structural analysis has succeeded. It is well known that SA can fail to find a DAE's true structure, thus producing an identically singular Jacobian: examples are the Campbell–Griepentrog Robot Arm [Campbell and Griepentrog 1995] and the Ring Modulator [Mazzia and Iavernaro 2003].

This difficulty has occurred surprisingly rarely in the applications we have met to date, but it should be catered for. To show success, it suffices to find *some* point that satisfies the equations for consistency and at which the System Jacobian $\mathbf{J}$ is nonsingular. It is reasonable to ask users to specify some initial data to this process, but maybe not as much, or not as carefully, as when specifying an IVP, and it is probably wise to make some random perturbations to such data to avoid isolated points of singularity. Preliminary tests on these lines are promising, and we aim to include such a method into DAESA when it has proven sufficiently robust.

ACKNOWLEDGMENTS

We acknowledge with thanks the support given to JDP by the Leverhulme Trust and the Engineering and Physical Sciences Research Council, both of the UK, and to GT and NSN by the Canadian Natural Sciences and Engineering Research Council and the McMaster Centre for Software Certification.

REFERENCES

BRENAN, K., CAMPBELL, S., AND PETZOLD, L. 1996. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations* second Ed. SIAM, Philadelphia.

CAMERON, I. AND GANI, R. 2011. *Product and Process Modelling: A Case Study Approach*. Elsevier Science, UK.

CAMPBELL, S. L. AND GRIEPENTROG, E. 1995. Solvability of general differential algebraic equations. *SIAM J. Sci. Comput. 16,* 2, 257–270.

CAO, Y. 2011. LAPJV, a MATLAB implementation of the Jonker-Volgenant algorithm for solving LAPs. `http://www.mathworks.com/matlabcentral/fileexchange/authors/22524`.

COLEMAN, T. F., EDENBRANDT, A., AND GILBERT, J. R. 1986. Predicting fill for sparse orthogonal factorization. *J. ACM 33,* 3, 517–532.

DAVIS, T. A. AND HU, Y. 2011. The university of Florida sparse matrix collection. *ACM Trans. Math. Softw. 38,* 1, 1:1–1:25.

DUFF, I. S., ERISMAN, A. M., AND REID, J. K. 1986. *Direct Methods for Sparse Matrices*. Oxford Science Publications. Clarendon Press, Oxford.

DUFF, I. S. AND GEAR, C. W. 1986. Computing the structural index. *SIAM Journal on Algebraic and Discrete Methods 7*, 594603.

DYNASYM AB. 2004. Dymola, dynamic modeling laboratory, user's manual. `http://www.inf.ethz.ch/personal/cellier/Lect/MMPS/Refs/Dymola5Manual.pdf`.

JONKER, R. AND VOLGENANT, A. 1987. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing 38*, 325–340. The assignment code is available at `www.magiclogic.com/assignment.html`.

MATTSSON, S. E. AND SÖDERLIND, G. 1993. Index reduction in differential-algebraic equations using dummy derivatives. *SIAM J. Sci. Comput. 14,* 3, 677–692.

MAZZIA, F. AND IAVERNARO, F. 2003. Test set for initial value problem solvers. Tech. Rep. 40, Department of Mathematics, University of Bari, Italy. `http://pitagora.dm.uniba.it/~testset/`.

NEDIALKOV, N. AND PRYCE, J. 2008–2009. DAETS user guide. Tech. rep., Department of Computing and Software, McMaster University, Hamilton, Ontario, Canada, L8S 4K1.

NEDIALKOV, N. S., PRYCE, J., AND TAN, G. 2012. DAESA — a Matlab tool for structural analysis of DAEs: Software.

NEDIALKOV, N. S. AND PRYCE, J. D. 2005. Solving differential-algebraic equations by Taylor series (I): Computing Taylor coefficients. *BIT 45*, 561–591.

NEDIALKOV, N. S. AND PRYCE, J. D. 2008. Solving differential-algebraic equations by Taylor series (III): The DAETS code. *JNAIAM 3,* 1–2, 61–80. ISSN 17908140.

PANTELIDES, C. C. 1988. The consistent initialization of differential-algebraic systems. *SIAM. J. Sci. Stat. Comput. 9*, 213–231.

POTHEN, A. AND FAN, C.-J. 1990. Computing the block triangular form of a sparse matrix. *ACM Transactions on Mathematical Software 16,* 4, 303–324.

PROCESS SYSTEMS ENTERPRISE LTD. 2004. gPROMS introductory user guide. `http://eng1.jcu.edu.au/Current%20Students/general/downloads/gPROMS/introductory_guide_231.pdf`.

PRYCE, J. D. 2001. A simple structural analysis method for DAEs. *BIT 41,* 2, 364–394.

TAN, G., NEDIALKOV, N. S., AND PRYCE, J. D. 2012a. Graph theory, irreducibility, and structural analysis of differential-algebraic equation systems. In preparation.

TAN, G., NEDIALKOV, N. S., AND PRYCE, J. D. 2012b. A simple method for quasilinearity analysis of differential-algebraic equation systems. In preparation.