A Simple Method for Quasilinearity Analysis of DAEs

Guangning Tan, Nedialko S. Nedialkov and John D. Pryce

Abstract We present a simple method for quasilinearity (QL) analysis of differential-algebraic equations (DAEs). It uses the sigma matrix and offsets computed by Pryce's structural analysis and determines if a DAE is QL in its leading derivatives. Our method is suitable for an implementation through either operator overloading or source code translation.

1 Introduction

We are interested in solving initial value problems in 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, and t is the time variable.

Based on Pryce's structural analysis (SA) [4], we solve (1) numerically using Taylor series, as implemented in the DAETS solver [2]. On each integration step, we compute Taylor coefficients for the solution up to some order, where we solve systems of equations for these coefficients in stages. Up to stage zero, a system can be linear or nonlinear in the variables being solved for, and after this stage, the systems are always linear.

G. Tan
Department of Computing and Software, McMaster University, Hamilton, Canada, e-mail: `tang4@mcmaster.ca`

N. S. Nedialkov
Department of Computing and Software, McMaster University, Hamilton, Canada, e-mail: `nedialk@mcmaster.ca`

J. Pryce
Cardiff School of Mathematics, Cardiff University, UK, e-mail: `prycejd1@Cardiff.ac.uk`

We present a simple method for deciding if such a system is linear in the unknown derivatives, respectively Taylor coefficients. We refer to such systems as quasilinear (QL). If the unknowns appear nonlinearly, we have a non-quasilinear (NQL) system. Such information is used to determine what solver to use and the minimum number of variables and derivatives of them that need initial conditions; for details see [5].

Sect. 2 summarizes Pryce's SA. Sect. 3 gives the definitions needed for our method. It is described in Sect. 4, and illustrated on an example in Sect. 5. Conclusions are in Sect. 6.

2 Summary of Pryce's SA

This SA [4] constructs for (1) an $n \times n$ *signature matrix* $\Sigma = (\sigma_{ij})$ such that

$$\sigma_{ij} = \begin{cases} \text{the highest order of the derivative to which } x_j \text{ occurs in } f_i\text{; or} \\ -\infty \text{ if } x_j \text{ does not occur in } f_i \,. \end{cases}$$

A highest value transversal (HVT) is a set of n positions (i, j) with one entry in each row and each column, such that the sum of these entries is maximized over all transversals. From Σ, we find a HVT and equation and variable offsets $\mathbf{c}$ and $\mathbf{d}$, respectively, which are non-negative integer n-vectors satisfying

$$d_j - c_i \geq \sigma_{ij} \quad \text{for all } i, j \text{ with equality on an HVT} \,.$$

When the SA succeeds [1, 4], using these offsets, we can determine structural index (which is a bound for the differentiation index, and often they are the same), degrees of freedom, and a solution scheme for computing derivatives of the solution.

They are computed in stages $k = k_d, k_d + 1, \ldots$, where $k_d = -\max_j d_j$. Denote

$$x_{J_k} = \left\{ x_j^{(d_j+k)} \mid d_j + k \geq 0 \right\}, \quad x_{J_{<k}} = \left\{ x_j^{(r)} \mid d_j + k > 0 \right\}, \quad \text{and}$$
$$f_{I_k} = \left\{ f_i^{(c_i+k)} \mid c_i + k \geq 0 \right\}.$$

At stage k, we solve a system of equations $f_{I_k}(t, x_{J_{<k}}, x_{J_k}) = 0$ for x_{J_k}, where $x_{J_{<k}}$ are computed at earlier stages. A system at stage $k = k_d, k_d + 1, \ldots, 0$ can be QL or NQL, while for stages $k > 0$ the systems are always linear.

Example 1. We show below for the simple pendulum (PEND), an index-3 DAE, the signature matrix and offsets. (The state variables are $x(t)$, $y(t)$, and $\lambda(t)$, G is gravity, and L is the length of the pendulum.) There are two HVTs, marked with $\bullet$ and $*$.

$$
\begin{aligned}
0 &= f_1 = x'' + x\lambda \\
0 &= f_2 = y'' + y\lambda - G \\
0 &= f_3 = x^2 + y^2 - L^2
\end{aligned}
\quad \rightarrow \quad
\Sigma = \begin{array}{c c c}
 & \begin{array}{ccc} x & y & \lambda \end{array} & c_i \\
\begin{array}{c} f_1 \\ f_2 \\ f_3 \end{array} & \begin{bmatrix} 2^\bullet & & 0^* \\ & 2^* & 0^\bullet \\ 0^* & 0^\bullet & \end{bmatrix} & \begin{array}{c} 0 \\ 0 \\ 2 \end{array} \\
d_j & \begin{array}{ccc} 2 & 2 & 0 \end{array} &
\end{array}
$$

The equations for stages $k = -2, -1, 0$ are

k	$f_{I_k}(t, x_{J_{<k}}, x_{J_k})$	$x_{J_{<k}}$	x_{J_k}	
-2	$f_3 = x^2 + y^2 - L^2$		x, y	NQL
-1	$f_3' = 2xx' + 2yy'$	x, y	x', y'	QL
0	$f_1 = x'' + x\lambda$ $f_2 = y'' + y\lambda - G$ $f_3'' = 2\big(xx'' + x'^2 + yy'' + y'^2\big)$	x, x', y, y'	x'', y'', λ	QL

Obviously, at $k = -2$ we have a NQL problem, and then two QL problems.

3 Quasilinearity at stage k

Definition 1. The system

$$f_{I_k}(t, x_{J_{<k}}, x_{J_k}) = 0 \tag{2}$$

is QL, if x_{J_k} appears linearly in it, and NQL otherwise.

Definition 2. A DAE is QL, if at stage $k = 0$, (2) is QL, and NQL otherwise.

Definition 3. Equation i at stage k is QL, if $f_i^{(k+c_i)} = 0$ is linear in the x_{J_k} occurring in it, and NQL otherwise.

If $c_i + k > 0$, $f_i^{(k+c_i)} = 0$ is always QL. For example, in PEND at stage $k = 1$, $f_3' = 2xx' + 2yy' = 0$ is QL in x' and y'.

At stage k, consider equations i for which $c_i + k = 0$. If each such $f_i = 0$ is QL, then (2) is QL; it is NQL if there is a NQL $f_i = 0$; cf. PEND at stage 0.

Therefore, to determine quasilinearity at stage k, we need to check for QL only the $f_i = 0$ for which $c_i + k = 0$.

4 Algorithm

For simplicity in our exposition, we consider the code list for evaluating the f_i's as consisting of assignment, unary, and binary operators. This is the case when executing the function for evaluating the DAE through operator overloading. Our algorithm consists of initialization and propagation of *offset* and *type* data through the code list as described below.

Initialization. We derive from Σ the $n \times n$ *offset* matrix $\Theta = (\theta_{ij})$ as

$$\theta_{ij} = \begin{cases} \sigma_{ij} & \text{if } \sigma_{ij} = d_j - c_i \\ +\infty & \text{otherwise} \,, \end{cases}$$

and derive from it the $n \times n$ *type* matrix $\mathtt{T} = (\mathtt{T}_{ij})$

$$\mathtt{T}_{ij} = \begin{cases} \mathtt{L} \quad \text{(Linear)} & \text{if } \theta_{ij} = 0 \\ \mathtt{U} \quad \text{(Undetermined)} & \text{if } 0 < \theta_{ij} < +\infty \\ \mathtt{C} \quad \text{(Constant)} & \text{if } \theta_{ij} = +\infty \,. \end{cases}$$

Then we associate with each x_j an *offset vector* $\gamma(x_j)$ being the jth column of Θ, and a *type vector* $\mathtt{T}(x_j)$ being the jth column of $\mathtt{T}$.

Propagation. We propagate these vectors through the code list of the DAE according to the following rules.

R1. If $v = +u$ or $v = -u$, then

$$\gamma(w) = \gamma(v) \quad \text{and} \quad \mathtt{T}(w) = \mathtt{T}(v) \,.$$

R2. If $u = g(v)$ is nonlinear, then

$$\gamma(u) = \gamma(v) \quad \text{and} \quad \mathtt{T}_i(u) = \begin{cases} \mathtt{N} \quad \text{(Nonlinear)} & \text{if } \mathtt{T}_i(v) = \mathtt{L} \\ \mathtt{T}_i(v) & \text{otherwise} \,. \end{cases}$$

R3. If $w = g(u,v)$, then for all $i = 1, \ldots n$,

$$\begin{aligned} &\gamma_i(w) = \min\{\, \gamma_i(u), \gamma_i(v) \,\} \quad \text{and} \\ &\mathtt{T}_i(w) = \begin{cases} \mathtt{N} & \text{if } \mathtt{T}_i(u) = \mathtt{T}_i(v) = \mathtt{L} \ \& \ g \text{ nonlinear} \\ \max\{\, \mathtt{T}_i(u), \mathtt{T}_i(v) \,\} & \text{otherwise} \,. \end{cases} \end{aligned}$$

Here we use the ordering

$$\mathtt{C} < \mathtt{U} < \mathtt{L} < \mathtt{N} \,.$$

R4. If $w = g(u,v)$, u is a constant or the time variable t, and v is not, then

$$\gamma(w) = \gamma(v) \quad \text{and} \quad \mathtt{T}(w) = \mathtt{T}(v) \,.$$

Similarly, when v is a constant or the time variable t, and u is not, then

$$\gamma(w) = \gamma(u) \quad \text{and} \quad \mathtt{T}(w) = \mathtt{T}(u) \,.$$

R5. If $v = d^p u/dt^p$, then

$$\gamma_i(v) = \gamma_i(u) - p \quad \text{and} \quad \mathtt{T}_i(v) = \begin{cases} \mathtt{L} & \text{if } \gamma_i(v) = 0 \\ \mathtt{U} & \text{if } 0 < \gamma_i(v) < +\infty \\ \mathtt{C} & \text{if } \gamma_i(v) = +\infty\,. \end{cases}$$

After executing the code list for an f_i, using R1-R5, $f_i = 0$ is QL at stage $k = -c_i$ if $\mathtt{T}_i(f_i) = \mathtt{L}$, and NQL if $\mathtt{T}_i(f_i) = \mathtt{N}$.

5 Example

We illustrate the above method on the following index-7 DAE

$$\begin{aligned} 0 &= f_1 = x'' + x\lambda \\ 0 &= f_2 = y'' + y\lambda + (x')^3 - G \\ 0 &= f_3 = x^2 + y^2 - L^2 \\ 0 &= f_4 = u'' + u\mu \\ 0 &= f_5 = (w''')^2 + w\mu - G \\ 0 &= f_6 = u^2 + w^2 - (L + c\lambda)^2 + \lambda'' \end{aligned} \tag{3}$$

(state variables are $x, y, \lambda, u, v,$ and w) derived from a two-coupled pendula problem, an index-5 DAE, with originally

$$f_2 = y'' + y\lambda - G, \quad f_5 = w'' + w\mu - G, \quad f_6 = u^2 + w^2 - (L + c\lambda)^2\,.$$

We wish to determine if the DAE (3) is QL; that is, if (3) is QL at stage $k = 0$. The corresponding matrices are

$$\begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ d_j \end{array}
\begin{array}{c} \begin{array}{cccccc|c} x & y & \lambda & u & w & \mu & c_i \end{array} \\
\left[\begin{array}{cccccc} 2 & & 0 & & & \\ 1 & 2 & 0 & & & \\ 0 & 0 & & & & \\ & & & 2 & & 0 \\ & & & & 3 & 0 \\ & & 2 & 0 & 0 & \end{array}\right] \begin{array}{c} 4 \\ 4 \\ 6 \\ 0 \\ 0 \\ 2 \end{array} \\
\begin{array}{cccccc} 6 & 6 & 4 & 2 & 3 & 0 \end{array} \end{array}
\qquad
\begin{array}{c} \begin{array}{cccccc|c} x & y & \lambda & u & w & \mu & c_i \end{array} \\
\left[\begin{array}{cccccc} 2 & & 0 & & & \\ & 2 & 0 & & & \\ 0 & 0 & & & & \\ & & & 2 & & 0 \\ & & & & 3 & 0 \\ & & 2 & 0 & & \end{array}\right] \begin{array}{c} 4 \\ 4 \\ 6 \\ 0 \\ 0 \\ 2 \end{array} \\
\begin{array}{cccccc} 6 & 6 & 4 & 2 & 3 & 0 \end{array} \end{array}
\qquad
\begin{array}{c} \begin{array}{cccccc} x & y & \lambda & u & w & \mu \end{array} \\
\left[\begin{array}{cccccc} \mathtt{U} & & \mathtt{L} & & & \\ & \mathtt{U} & \mathtt{L} & & & \\ \mathtt{L} & \mathtt{L} & & & & \\ & & & \mathtt{U} & & \mathtt{L} \\ & & & & \mathtt{U} & \mathtt{L} \\ & & \mathtt{U} & \mathtt{L} & & \end{array}\right] \\ \\ \end{array}$$

Σ, blanks denote $-\infty$ $\qquad$ Θ, blanks denote $+\infty$ $\qquad$ $\mathtt{T}$, blanks denote $\mathtt{C}$

Note that $d_1 - c_2 > \sigma_{2,1}$ and $d_5 - c_6 > \sigma_{6,5}$; hence these σ's do not appear in Θ.

Since $c_i = 0$ for equations 4 and 5, we need to examine only these two equations. (The remaining $f_i^{(c_i+k)} = 0$ are QL since $c_i > 0$ for $i = 1, 2, 3, 6$.)

Consider $f_5 = (w''')^2 + w\mu - G = 0$ with unknowns w''' and μ. We initialize

$$\gamma_5(w) = 3,\ \mathtt{T}_5(u) = \mathtt{U} \quad \text{and} \quad \gamma_5(\mu) = 0,\ \mathtt{T}_5(\mu) = \mathtt{L}$$

and propagate

code list	evaluates	$\gamma_5(v)$	$\mathtt{T}_5(v)$	applying
$v_4 = \mathrm{Dif}(w,3)$	$= w'''$	0	L	R5
$v_5 = v_4^2$	$= (w''')^2$	0	N	R2
$v_6 = w * \mu$	$= w\mu$	0	L	R3
$v_7 = v_5 + v_6$	$= (w''')^2 + w\mu$	0	N	R3
$f_5 = v_8 = v_7 - G$	$= (w''')^2 + w\mu - G$	0	N	R4

Since $\mathtt{T}_5(f_5) = \mathtt{N}$, f_5 is NQL. Hence this DAE is NQL.

6 Conclusion

We presented a simple method for quasilinearity analysis when solving a DAE by stages determined from Pryce's SA. Our method is implemented in the DAESA tool [3] for SA of DAEs and the DAETS solver [2]. In DAESA, we also construct a block-triangular form (BTF) of the DAE, and with this analysis, we determine the smallest number of variables and their derivatives that need initial values for a consistent initialization [3]. In DAETS, this method is used to select the appropriate solver when solving up to stage zero.

When applied block-wise to a BTF, our method considers variables that occur in positions outside diagonal blocks as constants. As a result, we need to set the corresponding off-diagonal entries in Θ to $+\infty$ and in $\mathtt{T}$ to $\mathtt{C}$. The propagation rules do not change.

The proof of correctness of our algorithm and a detailed description of how it works in the case of BTFs will be presented in a future work.

References

1. Nedialkov, N.S., Pryce, J.D.: Solving differential-algebraic equations by Taylor series (II): Computing the System Jacobian. BIT **47**(1), 121–135 (2007)
2. Nedialkov, N.S., Pryce, J.D.: Solving differential-algebraic equations by Taylor series (III): The DAETS code. JNAIAM **3**(1–2), 61–80 (2008)
3. Nedialkov, N.S., Pryce, J.D., Tan, G.: DAESA — a Matlab tool for structural analysis of DAEs: Software. ACM Transactions on Mathematical Software (2013). Accepted for publication
4. Pryce, J.D.: A simple structural analysis method for DAEs. BIT **41**(2), 364–394 (2001)
5. Pryce, J.D., Nedialkov, N.S., Tan, G.: DAESA — a Matlab tool for structural analysis of DAEs: Theory. ACM Transactions on Mathematical Software (2013). Accepted for publication