A NEW SOFTWARE PACKAGE FOR LINEAR DIFFERENTIAL–ALGEBRAIC EQUATIONS

PETER KUNKEL *, VOLKER MEHRMANN †, WERNER RATH †, AND JÖRG WEICKERT †

Abstract. We describe the new software package GELDA for the numerical solution of linear differential–algebraic equations with variable coefficients. The implementation is based on the new discretization scheme introduced in [21]. It can deal with systems of arbitrary index and with systems that do not have unique solutions or inconsistencies in the initial values or the inhomogeneity.

The package includes a computation of all the local invariants of the system, a regularization procedure and an index reduction scheme and it can be combined with every solution method for standard index 1 systems. Nonuniqueness and inconsistencies are treated in a least square sense.

We give a brief survey of the theoretical analysis of linear differential–algebraic equations with variable coefficients and discuss the algorithms used in GELDA. Furthermore, we include a series of numerical examples as well as comparisons with results from other codes, as far as this is possible.

Key words. Differential-algebraic equations, canonical forms, backward difference formulas, Runge-Kutta formulas, least square regularization, singular pencils, strangeness index.

AMS subject classification. 65L05

1. Introduction. We discuss the new software package GELDA for the numerical solution of linear differential–algebraic equations (DAE's) with variable coefficients

$$E(t)\dot{x}(t) = A(t)x(t) + f(t),\ t \in [\underline{t}, \bar{t}], \tag{1}$$

where $E, A \in C([\underline{t}, \bar{t}], \mathbf{C}^{n,n})$, $f \in C([\underline{t}, \bar{t}], \mathbf{C}^n)$ together with an initial condition

$$x(t_0) = x_0,\ t_0 \in [\underline{t}, \bar{t}],\ x_0 \in \mathbf{C}^n. \tag{2}$$

Here $C^\ell([\underline{t}, \bar{t}], \mathbf{C}^n)$ denotes the set of ℓ-times continuously differentiable functions from the interval $[\underline{t}, \bar{t}]$ to the n-dimensional complex vector space $\mathbf{C}^n$. (In this paper we discuss complex functions. In the case of real problems all the results are valid when $\mathbf{C}^n$, $\mathbf{C}^{n,n}$ are replaced by $\mathbf{R}^n$, $\mathbf{R}^{n,n}$, respectively.)

The theoretical analysis of such systems, regarding existence, uniqueness of solutions as well as consistency of initial conditions has been discussed in detail in [21, 22, 23]. We will survey only the relevant part of this work here to make the procedure that computes the invariants of the system transparent. The most important invariant in the analysis of linear DAE's is the so called *strangeness index*, which generalizes the differentiation index ([4, 6, 12]) for systems with undetermined components which occur, for example, in the solution of linear quadratic optimal control problems and differential–algebraic Riccati equations, see e.g. [19, 20, 28]. It is known that most of the standard integration methods for general DAE's require the system to have differentiation index not higher than one, which corresponds to a vanishing strangeness index, see [22]. If this condition is not valid or if the DAE has undetermined components, then standard methods as implemented in codes like DASSL of Petzold [29] or LIMEX of Deuflhard/Hairer/Zugck [8] often fail.

* Fachbereich Mathematik, Carl von Ossietzky Universität, Postfach 2503, D–26111 Oldenburg, Fed. Rep. Germany

† Fakultät für Mathematik, TU Chemnitz–Zwickau, D–09107 Chemnitz, Fed. Rep. Germany.

This work has been partially supported by the Deutsche Forschungsgemeinschaft under the grant no. Me 790/5-1 "Differentiell–algebraische Gleichungen".

The implementation of the new software package is based on the construction of the discretization scheme introduced in [21], which first determines all the local invariants and then transforms the system into a strangeness-free DAE with the same solution set. We will give a brief survey of this scheme. The resulting strangeness-free system may still have nonuniqueness in the solution set or inconsistent initial values or inhomogeneities. But this information is now available to the user and systems with such properties can be treated in a least squares sense.

In the case that the DAE is found to be uniquely solvable, we can compute a consistent initial value and apply the well-known integration schemes for DAE's. In our package we have implemented BDF methods [3] and Runge-Kutta-schemes [15, 16].

2. A brief survey of the basic results. In this section we give a brief survey of the results in [21, 22, 23]. The main results in these papers are the basis for the construction of the new software package.

We begin with two equivalence relations for pairs of matrix valued functions and matrix pairs, which play a central role in the theory of (1).

DEFINITION 1. Two pairs of matrix functions $(E_i(t), A_i(t))$ with $E_i, A_i \in C([\underline{t}, \bar{t}], \mathbf{C}^{n,n})$, $i = 1, 2$ are called *equivalent* if there exist $P \in C([\underline{t}, \bar{t}], \mathbf{C}^{n,n})$ and $Q \in C^1([\underline{t}, \bar{t}], \mathbf{C}^{n,n})$ with $P(t), Q(t)$ nonsingular for all $t \in [\underline{t}, \bar{t}]$ such that

$$(E_2(t), A_2(t)) = P(t)(E_1(t), A_1(t)) \begin{bmatrix} Q(t) & -\dot{Q}(t) \\ 0 & Q(t) \end{bmatrix}. \tag{3}$$

For the development of numerical methods one also needs a counterpart of this equivalence that can be obtained locally at a fixed point. At a fixed point $t \in [\underline{t}, \bar{t}]$ one can choose $Q(t)$ and $\dot{Q}(t)$ independently (see [13, 21]). This leads to the definition of local equivalence.

DEFINITION 2. Two pairs of matrices (E_i, A_i), $E_i, A_i \in \mathbf{C}^{n,n}$, $\quad i = 1, 2$ are called *locally equivalent* if there are matrices $P, Q, B \in \mathbf{C}^{n,n}$ with P, Q nonsingular such that

$$(E_2, A_2) = P(E_1, A_1) \begin{bmatrix} Q & -B \\ 0 & Q \end{bmatrix}. \tag{4}$$

Note that this is not a transformation that can be used in the DAE (1), since we cannot transform $x(t)$ and $\dot{x}(t)$ independently, but it is more appropriate than the usual equivalence for matrix pencils, see [11]. For the local equivalence (4) the following normal form is proved in [21]:

THEOREM 2.1. *Let $E, A \in \mathbf{C}^{n,n}$. Consider the following matrices and the spaces spanned by their columns:*

$$\begin{array}{ll} (a) & T \text{ basis of } \operatorname{kernel} E \\ (b) & Z \text{ basis of } \operatorname{corange} E = \operatorname{kernel} E^* \\ (c) & T' \text{ basis of } \operatorname{cokernel} E = \operatorname{range} E^* \\ (d) & V \text{ basis of } \operatorname{corange}(Z^*AT). \end{array} \tag{5}$$

Then the quantities (with the convention $\operatorname{rank} \emptyset = 0$*)*

$$\begin{array}{lll} (a) & r = \operatorname{rank} E & \text{(rank)} \\ (b) & a = \operatorname{rank}(Z^*AT) & \text{(algebraic part)} \\ (c) & s = \operatorname{rank}(V^*Z^*AT') & \text{(strangeness)} \\ (d) & d = r - s & \text{(differential part)} \\ (e) & u = n - r - a - s & \text{(undetermined part)} \end{array} \tag{6}$$

are well-defined and invariant under (4) and (E, A) is locally equivalent to the canonical form

$$(7)\qquad \left(\begin{bmatrix} I_s & 0 & 0 & 0 & 0 \\ 0 & I_d & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & I_a & 0 & 0 \\ I_s & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}\right) \begin{matrix} s \\ d \\ a \\ s \\ u \end{matrix}.$$

The characteristic values r, a, s, d, u are numerically computable via three rank decisions. There are essentially two robust methods to perform these decisions, the singular value decomposition and rank revealing QR-decompositions, e.g., [14]. In our code we currently use singular value decompositions, since this is the most reliable way to perform the required decisions, but note that numerical rank decisions are very difficult problems.

If we apply the reduction to the local canonical form (7) at every time t, we obtain functions

$$r, a, s : [\underline{t}, \bar{t}] \to \{0, \ldots, n\},$$

which in the current version of our algorithm are assumed to be constant, i.e.

$$(8)\qquad r(t) \equiv r,\ a(t) \equiv a,\ s(t) \equiv s.$$

The problem of analyzing the properties of systems where these quantities have discontinuities is not completely settled yet, partial results have been obtained in [7, 30, 24]. A general classification is currently under investigation.

For the equivalence relation (3) using $\dot{Q}$ instead of an arbitrary B one cannot eliminate as many elements of the pair as in (7). In [21] it is shown that for E, A in (1) sufficiently smooth and satisfying (8), equation (1) is equivalent to the system of differential–algebraic equations

$$\begin{array}{lll} (a) & \dot{x}_1(t) = A_{12}(t)x_2(t) + A_{14}(t)x_4(t) + A_{15}(t)x_5(t) + f_1(t) & (s \text{ equations}) \\ (b) & \dot{x}_2(t) = A_{24}(t)x_4(t) + A_{25}(t)x_5(t) + f_2(t) & (\text{d equations}) \\ (9)\ (c) & 0 = x_3(t) + f_3(t) & (a \text{ equations}) \\ (d) & 0 = x_1(t) + f_4(t) & (s \text{ equations}) \\ (e) & 0 = f_5(t) & (u \text{ equations}). \end{array}$$

Differentiating equation (9d) and inserting it in (9a), we obtain a new first equation

$$(10)\qquad 0 = A_{12}(t)x_2(t) + A_{14}(t)x_4(t) + A_{15}(t)x_5(t) + f_1(t) + \dot{f}_4(t).$$

Applying repeatedly the transformation to the form (9) and then the first equation to (10) one obtains an inductive definition of a sequence of pairs of matrix functions $(E_i(t), A_i(t))$ with normal form

$$(E_i(t), A_i(t)) \sim \left(\begin{bmatrix} I_{s_i} & 0 & 0 & 0 & 0 \\ 0 & I_{d_i} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & A_{12}^{(i)}(t) & 0 & A_{14}^{(i)}(t) & A_{15}^{(i)}(t) \\ 0 & 0 & 0 & A_{24}^{(i)}(t) & A_{25}^{(i)}(t) \\ 0 & 0 & I_{a_i} & 0 & 0 \\ I_{s_i} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}\right),$$

(11)

starting with $(E_0(t), A_0(t)) = (E(t), A(t))$ and $(E_{i+1}(t), A_{i+1}(t))$ is derived from $(E_i(t), A_i(t))$ by one step of this procedure.

Connected with this sequence, we then have sequences $r_i, a_i, s_i, d_i, u_i,\ i \in \mathbf{N}_0$ of nonnegative integers which are invariants for the given DAE, see [21].

Since $\operatorname{rank} E_{i+1}(t) = \operatorname{rank} E_i(t) - s_i$, there must be an index i such that $s_i = 0$, i.e. in this case the above process becomes stationary and the quantity

$$\mu = \min\{i \in \mathbf{N}_0 \mid s_i = 0\}, \tag{12}$$

is well-defined. As μ is characteristic for a given DAE of the form (1) with the above properties and gives the number of differentiations we must apply to reach a system with vanishing strangeness, μ is called the *strangeness index* of the DAE. Note that in contrast the *differentiation index* k is defined to be the number of differentiations one must apply to extract an ordinary differential equation such that every solution of the DAE is a solution of this ODE. In [21, 24] it has been shown that if both k and μ are well-defined, we then have $u_\mu = 0$, (i.e., no undetermined solution components) and $k = 0$ if $\mu = 0$ and $a_0 = 0$, or $k = \mu + 1$ otherwise.

For the numerical procedure we will also need the following quantities:

$$\begin{array}{lll} (a) & b_0 = a_0, & b_i = \operatorname{rank}\left(\begin{bmatrix} A_{14}^{(i-1)}(t) & A_{15}^{(i-1)}(t) \end{bmatrix}\right), \\ (b) & c_0 = a_0 + s_0, & c_i = \operatorname{rank}\left(\begin{bmatrix} A_{12}^{(i-1)}(t) & A_{14}^{(i-1)}(t) & A_{15}^{(i-1)}(t) \end{bmatrix}\right), \\ (c) & w_0 = u_0, & w_i = u_i - u_{i-1}, \qquad i = 1, \ldots, \mu, \end{array} \tag{13}$$

which relate to the invariants (see [21]) via

$$\begin{array}{ll} (a) & c_i = b_i + s_i, \qquad i = 0, \ldots, \mu \\ (b) & w_i = s_{i-1} - c_i, \quad i = 1, \ldots, \mu. \end{array} \tag{14}$$

The finite sequences $r_i, a_i, s_i,\ i \in \{0, \ldots, \mu\}$ (recall that d_i, u_i are dependent on these) lead to a general canonical form and the following existence and uniqueness results for (1), see [21].

THEOREM 2.2. *Let μ from (12) be well-defined for the differential-algebraic equation (1) and let $f \in C^{\mu+1}([\underline{t}, \bar{t}], \mathbf{C}^n)$. Then the following holds:*

i) Equation (1) is equivalent to a differential-algebraic equation of the form (with a notation adapted to a simplified block structure, i.e. different to (9), (10))

$$\begin{array}{ll} (a) & \dot{x}_1(t) = A_{13}(t)x_3(t) + f_1(t) \\ (b) & 0 = x_2(t) + f_2(t) \\ (c) & 0 = f_3(t) \end{array} \tag{15}$$

in the sense that their solutions are in one-to-one correspondence. The inhomogeneity is determined by $f^{(0)}, \ldots, f^{(\mu)}$. Furthermore d_μ, a_μ, u_μ are the number of differential, algebraic and undetermined components of the unknown x in (a), (b), (c) respectively.

ii) Equation (1) is solvable if and only if the u_μ functional consistency conditions

$$f_3(t) \equiv 0 \tag{16}$$

are satisfied.

iii) An initial condition (2) is consistent if and only if in addition the a_μ conditions

$$x_2(t_0) = -f_2(t_0) \tag{17}$$

hold.

iv) The initial value problem (1), (2) is uniquely solvable if again in addition we have

$$u_\mu = 0. \tag{18}$$

Otherwise, we can choose $x_3 \in C^1([\underline{t},\bar{t}], \mathbf{C}^{u_\mu})$ *arbitrarily.*

Note that for this theorem it is not really necessary to assume that the invariants are all constant, a weaker form with different assumptions is given in [24].

This existence and uniqueness result and the reduction procedure are of a theoretical nature, they are not feasible for numerical computation. The invariants and the properties of the system can, however, be nevertheless determined by a numerical algorithm. This algorithm is based on the computation of the local canonical form of different stages of a derivative array which is closely related to the derivative array introduced by Campbell [4], see also [3, 5].

To describe the array, we denote by a superscript $^{(i)}$ the i-th derivative and we define for sufficiently smooth $E(t), A(t)$ as in (1) and for $\ell = 0, \ldots, \mu$ the matrix functions $M_\ell(t), N_\ell(t)$ and the vector functions $z_\ell(t), g_\ell(t)$ blockwise by

$$\begin{aligned} (M_\ell)_{i,j} &:= \tbinom{i}{j} E^{(i-j)} - \tbinom{i}{j+1} A^{(i-j-1)},\ i,j = 0,\ldots,\ell \\ (N_\ell)_{i,j} &:= \begin{cases} A^{(i)} & \text{for } i = 0,\ldots,\ell,\ j = 0 \\ 0 & \text{otherwise.} \end{cases} \\ (z_\ell)_i &:= x^{(i)},\ i = 0,\ldots,\ell \\ (g_\ell)_i &:= f^{(i)},\ i = 0,\ldots,\ell \end{aligned} \tag{19}$$

such that (1) and its derivatives up to order ℓ can be written as

$$M_\ell(t)\dot{z}_\ell(t) = N_\ell(t) z_\ell(t) + g_\ell(t). \tag{20}$$

Note that we have used here the convention that $\binom{i}{j} = 0$ if $i < j$. The important observation made in [21] is that at a fixed point $\hat{t} \in [\underline{t},\bar{t}]$ the characteristic quantities corresponding to the large matrix pair $(M_\ell(\hat{t}), N_\ell(\hat{t}))$ are invariant under equivalence transformations to the original pair $(E(t), A(t))$, i.e., if the pairs $(E(t), A(t))$ and $(\tilde{E}(t), \tilde{A}(t))$ are equivalent via the transformation

$$\begin{aligned} \tilde{E}(t) &= P(t)E(t)Q(t) \\ \tilde{A}(t) &= P(t)A(t)Q(t) - P(t)E(t)\dot{Q}(t) \end{aligned} \tag{21}$$

as in (3) and if (M_ℓ, N_ℓ), $(\tilde{M}_\ell, \tilde{N}_\ell)$ are the corresponding large pairs constructed as in (19), then it follows for every fixed point $\hat{t} \in [\underline{t},\bar{t}]$ and for $\ell \in \mathbf{N}_0$ that

$$(\tilde{M}_\ell(\hat{t}), \tilde{N}_\ell(\hat{t})) = \Pi_\ell(\hat{t})(M_\ell(\hat{t}), N_\ell(\hat{t})) \begin{bmatrix} \Theta_\ell(\hat{t}) & -\Psi_\ell(\hat{t}) \\ 0 & \Theta_\ell(\hat{t}) \end{bmatrix}, \tag{22}$$

where

$$\begin{aligned} (\Pi_\ell)_{i,j} &:= \tbinom{i}{j} P^{(i-j)},\ (\Theta_\ell)_{i,j} := \tbinom{i+1}{j+1} Q^{(i-j)}, \\ (\Psi_\ell)_{i,j} &:= \begin{cases} Q^{(i+1)} & \text{for } i = 0,\ldots,\ell,\ j = 0, \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{23}$$

Thus, the characteristic quantities $(\tilde{r}_\ell, \tilde{a}_\ell, \tilde{s}_\ell, \tilde{d}_\ell, \tilde{u}_\ell)$ of the large pair $(M_\ell(\hat{t}), N_\ell(\hat{t}))$ are well-defined for equivalent pairs of matrix functions and each $\ell \in \mathbf{N}_0$. These

quantities are numerically computable via three rank decisions for each ℓ and each $\hat{t}$ and we can decide from these quantities on the size of μ and on the quantities $(r_i, a_i, s_i, d_i, u_i),\ i = 1, \ldots, \mu$ of the original pair $(E(t), A(t))$ at the point $t = \hat{t}$. In [21] the following relationship between the characteristic quantities $(\tilde{r}_\ell, \tilde{a}_\ell, \tilde{s}_\ell, \tilde{d}_\ell, \tilde{u}_\ell)$ of $(M_\ell(\hat{t}), N_\ell(\hat{t}))$ and $(r_i, a_i, s_i, d_i, u_i)$ of $(E(t), A(t))$ is proved:

$$
\begin{array}{ll}
\tilde{r}_\ell = (\ell+1)n - \sum_{i=0}^{\ell} c_i - \sum_{i=0}^{\ell} u_i, & \tilde{r}_\mu = (\mu+1)n - a_\mu - \sum_{i=0}^{\mu} u_i, \\
\tilde{a}_\ell = s_{\ell-1} - w_\ell - s_\ell = c_\ell - s_\ell, & \tilde{a}_\mu = c_\mu, \\
\tilde{s}_\ell = s_\ell + \sum_{i=0}^{\ell-1} c_i, & \tilde{s}_\mu = \sum_{i=0}^{\mu-1} c_i = a_\mu - c_\mu, \\
\tilde{d}_\ell = \tilde{r}_\ell - \tilde{s}_\ell = (\ell+1)n - c_\ell - \sum_{i=0}^{\ell} u_i, & \tilde{d}_\mu = (\mu+1)n - c_\mu - \sum_{i=0}^{\mu} u_i, \\
\tilde{u}_\ell = (\ell+1)n - \tilde{r}_\ell - \tilde{a}_\ell - \tilde{s}_\ell = \sum_{i=0}^{\ell} u_i, & \tilde{u}_\mu = \sum_{i=0}^{\mu} u_i, \\
\ell = 0, \ldots, \mu. &
\end{array} \tag{24}
$$

These formulas can be solved for the characteristic values $(r_i, a_i, s_i, d_i, u_i)$ of the original pair $(E(t), A(t))$, see [21].

$$
\begin{array}{rclrcl}
r_0 &=& \tilde{r}_0, & r_i &=& r_{i-1} - s_{i-1}, \\
a_0 &=& \tilde{a}_0, & s_i &=& \tilde{s}_i - v_{i-1}, \\
s_0 &=& \tilde{s}_0, & w_i &=& s_{i-1} - s_i - \tilde{a}_i, \\
d_0 &=& \tilde{d}_0, & u_i &=& w_i + u_{i-1}, \\
u_0 &=& \tilde{u}_0, & a_i &=& n - r_i - s_i - u_i, \\
w_0 &=& u_0, & d_i &=& r_i - s_i, \\
c_0 &=& a_0 + s_0, & c_i &=& s_{i-1} - w_i, \\
b_0 &=& a_0, & b_i &=& c_i - s_i, \\
v_0 &=& s_0 + a_0, & v_i &=& v_{i-1} + c_i,\ i = 1, \ldots, \mu.
\end{array} \tag{25}
$$

We have so far described the basis for our numerical procedure to determine the strangeness index μ and the other invariants of the system, i.e., for fixed $\hat{t}$ the large pairs $(M_\ell(\hat{t}), N_\ell(\hat{t}))$ are formed and the characteristic quantities are computed via the formulas (25). To do this we need to compute the invariants of $(M_\ell(\hat{t}), N_\ell(\hat{t}))$ under local equivalence for each ℓ. All the known numerical integration schemes for general DAE's essentially require the system to be strangeness free (or to have differentiation index 1). Thus the second task in a numerical procedure is the extraction of a strangeness free matrix pair $(\hat{E}(t), \hat{A}(t))$ and the corresponding DAE with local characteristic quantities $\hat{r} = d_\mu,\ \hat{a} = a_\mu,\ \hat{s} = 0$, the latter having the same solution structure as the original DAE (1). For reasons of numerical stability, the extraction procedure is exclusively based on the use of unitary projectors in the form of matrices with orthonormal columns.

In the following we assume that all the characteristic quantities of $(M_\ell(t), N_\ell(t)),\ \ell = 0, \ldots, \mu$ are independent of t and in addition we omit the index μ, i.e., $M_\mu =: M, N_\mu =: N,\ g_\mu =: g_0$, i.e., (8) holds for each ℓ.

It is shown in [21], that then there exists a smooth matrix valued function $Z_2(t)$ of size $((\mu+1)n, a_\mu)$, which we may assume w.l.o.g. to have orthonormal columns, such that

$$
Z_2(t)^* M(t) = 0, \quad \operatorname{rank}(Z_2(t)^* N(t) \begin{bmatrix} I_n \\ 0 \\ \vdots \\ 0 \end{bmatrix}) = a_\mu \tag{26}
$$

and there exists a smooth matrix valued function $T_2(t)$ of size $(n, d_\mu + u_\mu)$ with orthonormal columns, such that

$$Z_2(t)^* N(t) \begin{bmatrix} I_n \\ 0 \\ \vdots \\ 0 \end{bmatrix} T_2(t) = 0, \quad \operatorname{rank}(E(t)T_2(t)) = d_\mu . \tag{27}$$

Thus, there exists a smooth matrix function $Z_1(t)$ of size (n, d_μ), also with orthonormal columns, such that

$$\operatorname{rank}(Z_1(t)^* E(t)T_2(t)) = d_\mu . \tag{28}$$

Now instead of $Z_1(t)$ and $Z_2(t)$ we are only able to compute $Z_1(t)U_1(t)$ and $Z_2(t)U_2(t)$, where $U_1(t), U_2(t)$ are unitary but not necessarily smooth. The matrix pair that we obtain has the form

$$(\hat{E}(t), \hat{A}(t)) = \left(\begin{bmatrix} \hat{E}_1(t) \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \hat{A}_1(t) \\ \hat{A}_2(t) \\ 0 \end{bmatrix} \right), \tag{29}$$

where

$$\begin{array}{l} \hat{E}_1(t) = U_1(t)^* Z_1(t)^* E(t), \\ \hat{A}_1(t) = U_1(t)^* Z_1(t)^* A(t), \quad \hat{A}_2(t) = U_2(t)^* Z_2(t)^* N(t) \begin{bmatrix} I_n & 0 & \dots & 0 \end{bmatrix}^* . \end{array} \tag{30}$$

It can be shown that (29) has characteristic quantities

$$(\hat{r}, \hat{a}, \hat{s}, \hat{d}, \hat{u}) = (d_\mu, a_\mu, 0, d_\mu, u_\mu). \tag{31}$$

In particular (29) is strangeness free and since the unknown vector x is not transformed, the system has the same solution as the original equation.

Observe that $(\hat{E}(t), \hat{A}(t))$ may be non-smooth but that the product $\operatorname{diag}(U_1(t), U_2(t), I_{u_\mu})(\hat{E}(t), \hat{A}(t))$ is smooth. The numerical methods that we describe below are invariant under unitary transformations from the left, thus, since $\operatorname{diag}(U_1(t), U_2(t), I_{u_\mu})$ is unitary, we can assume w.l.o.g. that $(\hat{E}(t), \hat{A}(t))$ is smooth.

Note that for the numerical integration we need all functions only evaluated at discrete points, i.e., we never must integrate whole (smooth) functions.

3. Algorithms in GELDA. GELDA solves the system

$$E(t)\dot{x}(t) = A(t)x(t) + f(t) \tag{32}$$

of n differential-algebraic equations for x in a specified range of the independent variable t. Values $x(t_0) = x_0$ at the initial time t_0 can be given, but in contrast to standard methods, as implemented in DASSL [3], RADAU5 [16], or LIMEX [8], the initial values may be inconsistent. In the case of inconsistent initial values the code computes consistent initial values to start the integration.

The integration over the specified range of t values is done in a series of time steps. We use variable stepsize Runge–Kutta (RK) methods or backward differentiation formulas (BDF) methods which control the local error. Furthermore, the BDF method changes the order of the used formulas to compute the solution in a reliable and efficient way.

The integration algorithms used in GELDA are based on the theoretical results of Section 2. That is, we do not integrate the original DAE (1), instead we integrate a strangeness free DAE of the form

$$\begin{bmatrix} \hat{E}_1(t) \\ 0 \\ 0 \end{bmatrix} \dot{x}(t) = \begin{bmatrix} \hat{A}_1(t) \\ \hat{A}_2(t) \\ 0 \end{bmatrix} x(t) + \begin{bmatrix} \hat{f}_1(t) \\ \hat{f}_2(t) \\ \hat{f}_3(t) \end{bmatrix}, \tag{33}$$

with $\hat{f}_1(t) = U_1(t)^* Z_1(t)^* f(t)_1$, $\hat{f}_2(t) = U_2(t)^* Z_2(t)^* f(t)_2$ as it is obtained in (30). But we need not form this form globally, it is sufficient to obtain it locally at each integration step.

Note, that we set the third component of the inhomogeneity to zero, which yields a solvable system if the original inhomogeneity was inconsistent.

3.1. The Reduction Algorithm. Since the integration methods need the strangeness free DAE only at discrete points, we do not have to compute a global DAE of the form (33). Each time t_j the integration method requires (33), a subroutine is called, which computes in a first step the strangeness index μ and the quantities d_μ, a_μ, u_μ. This is done iteratively by building the large pairs M_ℓ, N_ℓ, $\ell = 0, 1, 2, \ldots$ corresponding to (32) at time t_j and computing the quantities in (25) until $s_\ell = 0$. Then, $\mu := \ell$ and we compute unitary projectors $Z_1(t_j), Z_2(t_j)$ and $T_2(t_j)$ as defined in (28), (26), (27) via three SVD's. Finally, these transformations are used to extract the strangeness free DAE (33).

Note, that we compute the strangeness index μ for each time t_j to detect whether the index μ or the characteristics d_μ, a_μ, u_μ change their value. If this happens, our method is not applicable and GELDA returns control to the calling program setting an error flag. For constant coefficient systems the characteristic quantities are constant in the whole interval. Even more, we can choose constant unitary projectors, so that the strangeness free DAE (33) has constant coefficients. Therefore, we need to compute the unitary projectors and $\hat{E}$, $\hat{A}$ only at the initial time t_0 and can use them in the whole interval.

The reduction process and the computation of the orthogonal transformation matrices is the most expensive part of the solution algorithm. We need $3(\mu+1)+2$ SVD's of different sizes for each time t_j we have to compute the strangeness free DAE.

3.2. Computing consistent initial conditions. The initial values $x(t_0) = x_0$ are allowed to be inconsistent, but before starting the integration we have to compute consistent initial values.

In GELDA, two approaches are implemented. The first possibility for computing consistent initial values x_0 is the following (see [21]). From the second block equation of (33) we get

$$\hat{A}_2(t_0)x(t_0) + \hat{f}_2(t_0) = 0.$$

For a given estimate $\tilde{x}_0 = x_0 + \delta$, the correction δ can be determined by the minimization problem

$$\|\delta\|_2 = \min!$$

subject to the constraint

$$\|\hat{A}_2(t_0)\delta - \hat{f}_2(t_0) - \hat{A}_2(t_0)\tilde{x}_0\|_2 = \min!$$

The solution of this minimization problem is well known (see [14]) and we get

$$\delta = \hat{A}_2(t_0)^+(\hat{A}_2(t_0)\tilde{x}_0 + \hat{f}_2(t_0)),$$

where $\hat{A}_2(t_0)^+$ is the Moore-Penrose pseudo-inverse of $\hat{A}_2(t_0)$.

If we are interested in finding arbitrary initial values x_0 this approach works well. But in some applications, e.g., multibody systems or electric circuits, we often know consistent initial values for the "differential variables", whereas initial values for the "algebraic and undetermined variables" are not known.

To derive a second method for the computation of consistent initial values, we start again with (33). Computing an LQ factorization of $\hat{E}_1(t_0)$, i.e., $\hat{E}_1(t_0) = [L\ 0]Q$, and setting $\hat{\dot{x}} = Q\dot{x}(t_0)$, $\hat{x} = Qx(t_0)$ we get

$$\begin{bmatrix} L & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\hat{x}}_1 \\ \dot{\hat{x}}_2 \end{bmatrix} = \begin{bmatrix} \hat{A}_{11} & \hat{A}_{12} \\ \hat{A}_{21} & \hat{A}_{22} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix} + \begin{bmatrix} \hat{f}_1(t_0) \\ \hat{f}_2(t_0) \\ 0 \end{bmatrix}.$$

¿From the second block row we then get

$$\hat{A}_2(t_0)x_0 + \hat{f}_2(t_0) = \hat{A}_{21}\hat{x}_1 + \hat{A}_{22}\hat{x}_2 + \hat{f}_2(t_0) = 0.$$

Let an estimate $\tilde{x}_0 = Q^*(\tilde{x}_1, \tilde{x}_2)$ be given. Since we want the "differential variables"$\hat{x}_1 = \tilde{x}_1$ to be fixed, we can determine the correction δ for the given estimate $\tilde{x}_2 = \hat{x}_2 + \delta$ by solving the minimization problem

$$\|\delta\|_2 = \min!$$

subject to the constraint

$$\|\hat{A}_{22}\delta - \hat{A}_2(t_0)\tilde{x} - \hat{f}_2(t_0)\|_2 = \min!,$$

i.e., $\delta = \hat{A}_{22}^+(\hat{A}_2(t_0)\tilde{x}(t_0) + \hat{f}_2(t_0))$. The corrected initial condition is then

$$x_0 = Q^* \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 - \delta \end{bmatrix}.$$

If the user wants GELDA to use the second method for the computation of consistent initial values, he has to set a flag, otherwise GELDA uses the first approach.

3.3. Integration Methods I: BDF. The BDF solver we use in GELDA is an adaption of the code implemented in DASSL (see [3, 29]). DASSL is a code for solving fully implicit (differential) index–1 DAE's of the form

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

We will briefly discuss the algorithms and strategies in the BDF code of GELDA. For a detailed discussion about the used order and stepsize selection strategies the reader is referred to [3].

Description of the method. Let x_{N-i} be approximations to the solution $x(t_{N-i})$, $i = 0, 1, \ldots, k$ of the DAE (33), where k is the order of the BDF method we plan to use. We are interested in an approximation for the solution $x(t_{N+1})$ at

time t_{N+1}. First, we compute an approximation to $x(t_{N+1})$ by evaluating a predictor polynomial at time t_{N+1}. The predictor polynomial ω^P_{N+1} is defined by

$$\omega^P_{N+1}(t_{N-i}) = x_{N-i}, \quad i = 0, 1, \ldots, k, \tag{34}$$

i.e., it interpolates at the last $k+1$ points.

A first approximation of the solution and its first derivative at t_{N+1} are then given by

$$\begin{aligned} x^P_{N+1} &= \omega^P_{N+1}(t_{N+1}) \\ \dot{x}^P_{N+1} &= \dot{\omega}^P_{N+1}(t_{N+1}). \end{aligned} \tag{35}$$

The approximation x_{N+1} to the solution that we accept is the solution of the corrector formula. As in DASSL, we use the fixed leading coefficient form of the k–th order BDF method. The approximation x_{N+1} is implicitly defined by

$$\begin{aligned} \omega^C_{N+1}(t_{N+1}) &= x_{N+1} \\ \omega^C_{N+1}(t_{N+1} - ih_{N+1}) &= \omega^P_{N+1}(t_{N+1} - ih_{N+1}), \qquad 1 \le i \le k \\ \hat{E}(t_{N+1})\dot{\omega}^C_{N+1}(t_{N+1}) &= \hat{A}(t_{N+1})\omega^C_{N+1}(t_{N+1}) + \hat{f}(t_{N+1}), \end{aligned} \tag{36}$$

where ω^C_{N+1} and $\dot{\omega}^C_{N+1}$ are the corrector polynomial and its derivative.

The predictor and corrector polynomial are represented in terms of modified divided differences of x. At each step we have to update the following quantities

$$\begin{gathered} \psi_i(N+1) = \sum_{j=0}^{i-1} h_{N+1-j} = t_{N+1} - t_{N+1-i}, \quad i \ge 1; \\ \beta_1(N+1) = 1; \quad \beta_i(N+1) = \prod_{j=1}^{i-1} \tfrac{\psi_j(N+1)}{\psi_j(N)}, \quad i \ge 1; \\ \phi_1(N) = x_N; \quad \phi_i(N) = [x_N, x_{N-1}, \ldots, x_{N-i+1}] \prod_{j=1}^{i-1} \psi_j(N), \quad i \ge 1; \\ \phi^*_i(N) = \beta_i(N+1)\phi_i(N), \quad i \ge 1; \\ \sigma_1(N+1) = 1; \quad \sigma_i(N+1) = h^i_{N+1}(i-1)! \prod_{j=1}^{i} \tfrac{1}{\psi_j(N+1)}, \quad i \ge 1; \\ \alpha_i(N+1) = h_{N+1}/\psi_i(N+1), \quad i \ge 1; \\ \alpha_s = -\sum_{j=1}^{k} \tfrac{1}{j}; \\ \alpha^0(N+1) = -\sum_{j=1}^{k} \alpha_j(N+1); \\ \gamma_1(N+1) = 0; \quad \gamma_i(N+1) = \gamma_{i-1}(N+1) + \alpha_{i-1}(N+1)/h_{N+1}, \quad i \ge 1. \end{gathered} \tag{37}$$

The divided differences are defined recursively by

$$\begin{aligned} [x_N] &= x_N \\ [x_N, x_{N-1}, \ldots, x_{N-k}] &= \tfrac{[x_N, x_{N-1}, \ldots, x_{N-k+1}] - [x_{N-1}, x_{N-2}, \ldots, x_{N-k}]}{t_N - t_{N-k}} \end{aligned}$$

and the predictor polynomial is given by

$$\omega^P_{N+1}(t) = \sum_{i=1}^{k+1} \prod_{j=1}^{i-1} (t - t_{N+1-j})[x_N, x_{N-1}, \ldots, x_{N+1-i}]. \tag{38}$$

After evaluating $\omega_{N+1}^P(t)$ at t_{N+1} and using notation (37), we get the predicted value

$$x_{N+1}^P = \sum_{i=1}^{k+1} \phi_i^*(N). \tag{39}$$

For the derivative of (38) at t_{N+1} we find

$$\dot{x}_{N+1}^P = \sum_{i=1}^{k+1} \gamma_i(N+1)\phi_i^*(N). \tag{40}$$

Finally, from (36) we find after some manipulations

$$\hat{E}(t_{N+1})[\dot{x}_{N+1}^P - \frac{\alpha_s}{h_{N+1}}(x_{N+1} - x_{N+1}^P)] = \hat{A}(t_{N+1})x_{N+1} + \hat{f}(t_{N+1}). \tag{41}$$

Note that (41) is linear in x_{N+1}. Therefore, we can solve for x_{N+1} and get that

$$\begin{aligned} &[\tfrac{\alpha_s}{h_{N+1}}\hat{E}(t_{N+1}) + \hat{A}(t_{N+1})]x_{N+1} = \\ &\qquad \hat{E}(t_{N+1}) \sum_{i=1}^{k+1} \left(\gamma_i(N+1) + \tfrac{\alpha_s}{h_{N+1}}\right) \phi_i^*(N) - \hat{f}(t_{N+1}). \end{aligned} \tag{42}$$

If $u_\mu \equiv 0$ then (42) has a unique solution x_{N+1}. For $u_\mu \not\equiv 0$ the solution of (42) is not unique. Therefore we set $x_{N+1} := x_N + \Delta x_{N+1}$ and use a least squares approach to compute a unique minimal norm solution Δx_{N+1} (see Section 3.5 for details).

Note that in both cases $u_\mu \equiv 0$ or $u_\mu \not\equiv 0$, the solution of (42) is invariant under unitary transformations from the left.

After a successful step we use the polynomial

$$\omega_{N+1}^I(t) = \sum_{i=0}^{k+1} \prod_{j=0}^{i-1} (t - t_{N-j})[x_N, x_{N-1}, \ldots, x_{N-i}]$$

to interpolate at times t between t_N and t_{N+1}. Here, k is the order of the method used to integrate form t_N to t_{N+1}.

Order and stepsize control. The BDF code changes the order of the method and the stepsize to compute accurate approximations in an efficient and reliable way.

The error is estimated by

$$\text{err} = \max(M_1, M_2)\|x_{N+1} - x_{N+1}^P\| \leq 1.0, \tag{43}$$

where

$$\begin{aligned} M_1 &= \alpha_{k+1}(N+1) \\ M_2 &= |\alpha_{k+1}(N+1) + \alpha_s - \alpha^0(N+1)|. \end{aligned} \tag{44}$$

Two types of errors have been considered. For output purpose we need to compute the solution at a point t^* between the meshpoints $t_N \leq t^* \leq t_{N+1}$, this is done by interpolation, i.e., evaluating $\omega_{N+1}^I(t)$ at t^*. The term M_1 in (44) comes from the estimation of the principal part of the interpolation error.

The other part of the error is the local truncation error of the method. This is the error by which the solution of the DAE fails to satisfy the BDF formula. M_2 in (44)

is due to the estimation of this truncation error. Note, that this type of error is also introduced in the interpolation, since we do not interpolate the exact solution.

The estimate of the local truncation error we use is asymptotically correct for fixed stepsizes (see [3] and the references therein). Therefore, the stepsize and order selection strategies prefers sequences of constant stepsize and order.

A step is accepted if (43) is satisfied, otherwise it is rejected. After each accepted or rejected step the code must decide which order is to be used in the next step. For this, the code uses the estimates

$$
\begin{aligned}
\text{TERKM2} &= \|(k-1)\sigma_{k-1}(N+1)\phi_k(N+1)\| \approx \|h^{k-1}x^{(k-1)}\| \\
\text{TERKM1} &= \|k\sigma_k(N+1)\phi_{k+1}(N+1)\| \approx \|h^k x^{(k)}\| \\
\text{TERK} &= \|(k+1)\sigma_{k+1}(N+1)\phi_{k+2}(N+1)\| \approx \|h^{k+1}x^{(k+1)}\| \\
\text{TERKP1} &= \|(k+2)\sigma_{k+2}(N+1)\phi_{k+3}(N+1)\| \approx \|h^{k+2}x^{(k+2)}\|
\end{aligned}
\tag{45}
$$

of order $k-2, k-1, k$ and $k+1$. The estimate TERKP1 of order $k+1$ is computed only after $k+1$ steps of constant stepsize.

At the beginning of the integration, the code starts with an initialization phase, in which order and stepsize are increased in each step. Later, the order is increased or decreased if TERKM2, TERKM1, TERK and TERKP1 form an increasing or decreasing sequence.

Finally, the code has to decide which stepsize should be used for the next step. The new stepsize rh_{N+1} is chosen so that the expected error is about one half of the desired integration error tolerance. If EST is the error estimation of order k, which should be used in the next step, r is given by

$$
r = (2.0\ \text{EST})^{-1/(k+1)}. \tag{46}
$$

Note, that EST can be obtained by (45), but has to be scaled by the error constant $1/(k+1)$.

The stepsize is increased after a successful step only if it can be doubled and then it is doubled. If the stepsize must be decreased after a successful step, the stepsize is decreased by at least $r = 0.9$, and at most $r = 0.5$. After a rejected step r lies between $r = 0.9$ and $r = 0.25$. Moreover, after three consecutive rejected steps the order is reduced to one and the stepsize is reduced by $r = 0.25$ until a step is accepted.

The norm we use in the BDF code is a weighted root mean square norm. Precisely, it is defined by

$$
\|v\| = \left(\frac{1}{N} \sum_{i=1}^{N} \left(\frac{v_i}{\text{WT}_i} \right)^2 \right)^{1/2}, \tag{47}
$$

where $\text{WT}_i = \text{RTOL}_i |x_i| + \text{ATOL}_i$ and RTOL, ATOL are the relative and absolute integration tolerances the user must provide.

3.4. Integration Methods II: RK. The Runge–Kutta branch of the integration schemes was adopted from the method RADAU5 of Hairer and Wanner [16] . This is a 3–stage implicit Runge–Kutta method of the RadauIIA type of fifth order for the solution of DAE's of the form $M \cdot \dot{Y} = f(t, Y)$. At each point where matrix evaluations are required - i.e. three times in every time step - the process of index reduction is carried out (see subsection 3.1). Therefore, the Runge–Kutta solver always treats strangeness free systems.

Description of the method. Since the coefficients of the DAE (33) are generally time–dependent, RADAU5 is not directly applicable to this case. Therefore, we make use of the following transformation: We define $y := \dot{x}$ and consider the enlarged system

$$\begin{aligned} \dot{x} &= y \\ 0 &= -\hat{E}y + \hat{A}x + \hat{f} \end{aligned} \tag{48}$$

which represents a linear semi–explicit DAE of strangeness index 1 with constant singular matrix $M = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}$ multiplying the derivative, i.e. RADAU5 can be applied to (48). Writing down the RK scheme for (48) and defining (t_0, x_0) as initial point of the current time step, x_1 as the new solution in this time step, a_{ij}, b_j, c_i as coefficients of the RK scheme, we obtain after some transformations the following equations fixing the method:

$$z_i = h \sum_{j=1}^{3} a_{ij}\, y_j, \qquad i = 1,2,3; \tag{49}$$

$$\hat{E}(t_0 + c_i h) y_i = \hat{A}(t_0 + c_i h)(x_0 + z_i) + \hat{f}(t_0 + c_i h), \qquad i = 1,2,3; \tag{50}$$

$$x_1 = x_0 + h \sum_{j=1}^{3} b_j\, y_j. \tag{51}$$

Equations (49), (50) and (51) are the same as would be obtained by directly applying the RK scheme to the original equation (33).
Since the method is stiffly accurate (see [16]), i.e. $b_j = a_{3j}$, the new solution can be obtained by

$$x_1 = x_0 + z_3. \tag{52}$$

Inserting (49) into (50) (with the abbreviations $t_j := t_0 + c_j h$) yields

$$\hat{E}(t_i) y_i = h\hat{A}(t_i) \sum_{j=1}^{3} a_{ij}\, y_j + \hat{A}(t_i) x_0 + \hat{f}(t_i),$$

so that our Runge–Kutta system reads

$$\begin{bmatrix} \hat{E}(t_1) - ha_{11}\hat{A}(t_1) & -ha_{12}\hat{A}(t_1) & -ha_{13}\hat{A}(t_1) \\ -ha_{21}\hat{A}(t_2) & \hat{E}(t_2) - ha_{22}\hat{A}(t_2) & -ha_{23}\hat{A}(t_2) \\ -ha_{31}\hat{A}(t_3) & -ha_{32}\hat{A}(t_3) & \hat{E}(t_3) - ha_{33}\hat{A}(t_3) \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \tag{53}$$

$$\begin{bmatrix} \hat{A}(t_1)x_0 + \hat{f}(t_1) \\ \hat{A}(t_2)x_0 + \hat{f}(t_2) \\ \hat{A}(t_3)x_0 + \hat{f}(t_3) \end{bmatrix}.$$

Although (53) represents a linear system for the unknowns y_1, y_2, y_3 (in contrast to the system of RADAU5, where the solution vector is composed of the z_i), the system

matrix in (53) is the same as in the Newton iterations of RADAU5 — provided that $\hat{E}$ and $\hat{A}$ do not vary in t_1, t_2 and t_3. Therefore, in the case of constant coefficients, we can make use of the following strategy as applied in RADAU5: For $\hat{E}(t_1) = \hat{E}(t_2) = \hat{E}(t_3) =: \hat{E}$ and $\hat{A}(t_1) = \hat{A}(t_2) = \hat{A}(t_3) =: \hat{A}$, the system (53) can be written as

$$(I \otimes \hat{E} - h\mathcal{A} \otimes \hat{A})Y = B \tag{54}$$

with

$$\mathcal{A} = (a_{ij})_{i,j=1}^{3} \text{ the Runge–Kutta matrix,}$$

$$Y = [y_1, y_2, y_3]^T \text{ and}$$

$$B = \begin{bmatrix} \hat{A}x_0 + \hat{f}(t_1) \\ \hat{A}x_0 + \hat{f}(t_2) \\ \hat{A}x_0 + \hat{f}(t_3) \end{bmatrix}.$$

In (54) we have used the Kronecker product of two matrices which is defined for $C \in \mathbf{R}^{n,m}$, $D \in \mathbf{R}^{k,\ell}$ by the block matrix

$$C \otimes D = \begin{bmatrix} c_{11}D & \cdots & c_{1m}D \\ \vdots & \ddots & \vdots \\ c_{n1}D & \cdots & c_{nm}D \end{bmatrix} \in \mathbf{R}^{nk,m\ell}.$$

$\mathcal{A}^{-1}$ has one real eigenvalue γ and one complex conjugate eigenvalue pair $\alpha \pm i\beta$ and can be transformed into block diagonal form:

$$\mathcal{A}^{-1} = T\Lambda T^{-1} \text{ with } \Lambda = \begin{bmatrix} \gamma & 0 & 0 \\ 0 & \alpha & -\beta \\ 0 & \beta & \alpha \end{bmatrix}.$$

Premutiplying (54) by $(h\mathcal{A})^{-1} \otimes I$ and defining $V := (T^{-1} \otimes I)Y$, we obtain the system

$$(h^{-1}\Lambda \otimes \hat{E} - I \otimes \hat{A})V = (h^{-1}\Lambda \otimes I)(T^{-1} \otimes I)B \tag{55}$$

with system matrix

$$\begin{bmatrix} h^{-1}\gamma\hat{E} - \hat{A} & 0 & 0 \\ 0 & h^{-1}\alpha\hat{E} - \hat{A} & -h^{-1}\beta\hat{E} \\ 0 & h^{-1}\beta\hat{E} & h^{-1}\alpha\hat{E} - \hat{A} \end{bmatrix}.$$

Instead of the decomposition of a $(3n \times 3n)$–system, we have to treat only one real $(n \times n)$ and one complex $(n \times n)$–system. This means a reduction of the amount of work to $5 \cdot O(n^3)$, compared to $27 \cdot O(n^3)$ for the $(3n \times 3n)$–system.
The question arises whether this technique can be applied in some way to the case of time–dependent coefficients, too. The idea is to transform (53) into a system of a simplified Newton method, holding $\hat{E}$ and $\hat{A}$ in the system matrix constant with respect to each single time step. Unfortunately, the process of index reduction does not supply continuous sequences $(\hat{E}(t_i), \hat{A}(t_i))$ since these matrices may be pre–muliplied

by unitary matrices. This effects that the pair $(\hat{E}(t_0), \hat{A}(t_0))$ at the left hand side of the Newton system may not be close to the corresponding pair $(\hat{E}(t_i), \hat{A}(t_i))$ at the right hand side, even if t_i is close to t_0. Consequently, in the case of time–varying coefficients, it is not possible to make use of the special structure of the matrix in (53), and we have to decompose the whole $(3n \times 3n)$–matrix at each time step.
For both strategies, the system matrix is nonsingular for sufficiently small h and $u_\mu \equiv 0$ (i.e. no undetermined components appear in the pencil $(\hat{E}, \hat{A})$). For $u_\mu \not\equiv 0$ we compute a least squares solution. The solution process is described in Section 3.5.

Error estimation. The estimator for the local error of RADAU5, developed for an embedded pair of methods, reads

$$\text{(56)} \quad \text{err} = ((h\gamma_0)^{-1}M - J)^{-1}[\tilde{f}(t_0, x_0, y_0) + (h\gamma_0)^{-1}M(e_1 Z_1 + e_2 Z_2 + e_3 Z_3)],$$

where $\tilde{f}$ stands for the right–hand side of the enlarged system (48), $\gamma_0 = \gamma^{-1}$ and the first n components of the Z_i coincide with the z_i of (49). Because of the special structure of $M = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}$, we only have to consider these first n components of the Z_i and need not know what the other ones represent. With the Jacobian $J = \begin{bmatrix} 0 & I \\ \hat{A}(t_0) & -\hat{E}(t_0) \end{bmatrix}$, we have

$$((h\gamma_0)^{-1}M - J)^{-1} = \begin{bmatrix} h^{-1}\gamma I & -I \\ -\hat{A}(t_0) & \hat{E}(t_0) \end{bmatrix}^{-1},$$

which is found to be

$$\begin{bmatrix} (h^{-1}\gamma\hat{E} - \hat{A})^{-1} & 0 \\ 0 & (h^{-1}\gamma\hat{E} - \hat{A})^{-1} \end{bmatrix} \begin{bmatrix} \hat{E}(t_0) & I \\ \hat{A}(t_0) & h^{-1}\gamma I \end{bmatrix}.$$

The second block of $\tilde{f}$ equals $-\hat{E}(t_0)y(t_0) + \hat{A}(t_0)x(t_0) + f(t_0)$ which is the residual of the previous step. Therefore, we can exclude these components in (56) and have as estimation for the local error

$$\text{err} = \begin{bmatrix} (h^{-1}\gamma\hat{E} - \hat{A})^{-1} & 0 \\ 0 & (h^{-1}\gamma\hat{E} - \hat{A})^{-1} \end{bmatrix} \begin{bmatrix} \hat{E}(t_0)[y_0 + h^{-1}\gamma(e_1 z_1 + e_2 z_2 + e_3 z_3)] \\ \hat{A}(t_0)[y_0 + h^{-1}\gamma(e_1 z_1 + e_2 z_2 + e_3 z_3)] \end{bmatrix}.$$

In the case of constant coefficients, the LU decomposition of $(h^{-1}\gamma\hat{E} - \hat{A})^{-1}$ is already known from the previous work since this matrix occurs in the left upper block of the system matrix, i.e. it is the matrix of the real $(n \times n)$–system. For y_0 we take y_3 of the previous step which is an approximation to the derivative $\dot{x}$ in t_0. Note that in the error norm the $(n+1)$st through $(2n)$th component of err must be multiplied by h since those components represent the index–2 part with respect to the enlarged system (48).
We only need an error estimation for x, i.e. for the x–component of (48), for which the local error is $O(h^5)$ (see [16]). Hence, the fifth order accuracy for the local error is preserved.

Stepsize control. The step sizes are determined from the local error in the same way as in the code RADAU5. The new step size is defined by

$$h_{\text{new}} = fac \cdot h_{\text{old}} \left(\frac{1}{\|\text{err}\|} \right)^{0.25},$$

where we use the same norm as in (47). In the current version we always set $fac = 0.9$ as RADAU5 would do in the case of one Newton iteration. This is motivated from the fact that we are solving linear systems.

The relative and absolute error tolerances can be chosen according to the desired exactness. The current step is accepted if $\|\mathrm{err}\| < 1$, otherwise rejected.

3.5. Linear Algebra. In GELDA we find many basic and advanced linear algebra operations. These computations are performed by calls to the Basic Linear Algebra Subprograms (BLAS) [26, 9] or the Linear Algebra Package (LAPACK) [1], which itself is based on BLAS.

Two types of BLAS routines are called. For vector–vector operations, such as $y \leftarrow \alpha x + y$, we use the Level 1 BLAS [26] and for matrix–vector operations, such as $y \leftarrow \alpha Ax + y$ we use the Level 2 BLAS [9].

Besides these basic operations, we have to solve linear systems of the form

$$Ax = b. \tag{57}$$

Depending on u_μ, the number of unknown components of the DAE, the matrix A is invertible or not.

For $u_\mu \equiv 0$ we have a non–singular A and the system (57) is solved by LAPACK linear system solvers. We use xGEEQU[1] and xLAQGE to equilibrate A, xGETRF to compute the LU factorization and xGETRS to solve the system.

For $u_\mu \not\equiv 0$ the linear system (57) is not uniquely solvable. In this case we compute a least squares solution, that is, we replace (57) by

$$\|x\|_2 = \min! \tag{58}$$

subject to the constraint

$$\|Ax - b\|_2 = \min!.$$

The solution to (58) is obtained by the LAPACK linear least squares solver xGELSS, which is based on the singular value decomposition (SVD) of A. This code is also used for the computation of consistent initial values.

4. Using GELDA. Copies of GELDA can be obtained by contacting Volker Mehrmann by email at *na.mehrmann@na-net.ornl.gov*.

A detailed description of the code and how to use it is given in [25].

5. Comparison of GELDA with other software packages. In this section we describe the results that the double precision version of GELDA yielded for several test problems which cover a wide range of applications. We compare the behaviour of the BDF– and RK–solver in GELDA with other DAE solvers as DASSL [29] and RADAU5 [16]. For constrained multibody systems we additionally use the codes MEXX [27] and ODASSL [10], which are designed to solve this special type of DAE's. The computations have been carried out on a Hewlett Packard 9000/720 workstation.

EXAMPLE 1. Our first example is the ODE $\dot{x}(t) = Ax(t)$, where $A = (a_{k,l})_{k,l=1}^{n}, n = m^2$ for $m \in \mathbf{N}$ and

$$\begin{array}{lcrcl}
a_{k,k} & = & -2 & : & k = i + (j-1)m;\ i,j = 1,\ldots,m; \\
a_{k,k-1} & = & 1 & : & k = i + (j-1)m;\ i = 2,\ldots,m;\ j = 1,\ldots,m; \\
a_{k,k-m} & = & 1 & : & k = i + (j-1)m;\ i = 1,\ldots,m;\ j = 2,\ldots,m; \\
a_{k,l} & = & 0 & : & \text{otherwise.}
\end{array}$$

[1] Depending on the data type, the first letter "x" of a subroutine must be replaced by "S" for the single or "D" for the double precision version of GELDA.

Using the initial condition $x(0) = [1, 0, \ldots, 0]$, the solution of this ODE is

$$x_k(t) = \frac{1}{(i-1)!\,(j-1)!} t^{i+j-2} e^{-2t} \quad : \quad k = i + (j-1)m,\ i, j = 1, \ldots, m.$$

As output points we used $t_{\text{out}} = 0.01, 0.1, 1, 10, 100$.

The origin of this problem is the method of lines approach for the 2–D advection partial differential equation (PDE) $u_t = -u_x - u_y$, where the spatial derivatives are discretized by finite differences to convert the PDE into an ODE.

EXAMPLE 2. This example from [17, 18] comes from a control problem of a three–link planar manipulator. For a linearized model we get a closed loop tracking system of the form

$$\begin{bmatrix} E & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \dot{x}(t) \\ \dot{x_c}(t) \end{bmatrix} = \begin{bmatrix} A + BK_x & BK_c \\ -B_c C & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ x_c(t) \end{bmatrix} + \begin{bmatrix} 0 \\ B_c r_s \end{bmatrix}, \tag{59}$$

which is a linear constant coefficient DAE with $x(t) \in \mathbf{R}^8$ and $x_c(t) \in \mathbf{R}^3$. In the interval $[0, 30]$ the tracking signal is $r_s(t) = \begin{bmatrix} r_{s1}(t) & 0 & 0 \end{bmatrix}$, with

$$r_{s1}(t) = \begin{cases} 0.5 & : \quad t \in [0,5), [10,15), [20,30] \\ -0.5 & : \quad t \in [5,10), [15,20) \end{cases}$$

and we chose the initial condition

$$\begin{bmatrix} x(0)^T & x_c(0)^T \end{bmatrix} = \begin{bmatrix} 0 & -0.5 & 0 & 0 & 0 & -0.3 & 0 & 20 & 0 & 0 & 0 \end{bmatrix}.$$

This initital value is inconsistent and GELDA computes the consistent initial condition

$$\begin{bmatrix} x(0)^T & x_c(0)^T \end{bmatrix} = \begin{bmatrix} 0 & -0.5 & 0 & 0 & 0 & -0.3 & 0 & -0.8655 & 0.3597 & 0 & 0 \end{bmatrix}.$$

As output points we use $t_{\text{out}} = 5, 10, 15, 20, 30$, i.e. the points where the inhomogeneity is not continous. Furthermore, we stop the integration at the output points and restart the code for the next interval.

Finally, since the exact solution of this problem is not known, we computed a reference solution at the output points in quadruple precision using different DAE codes with tight tolerances.

EXAMPLE 3. This example is a model of a constrained multibody system written in the Euler–Lagrange equations

$$\begin{aligned} \dot{p} &= v \\ M(p,t)\dot{v} &= f(p,v,t) - G^T(p,t)\lambda \\ 0 &= g(p,t). \end{aligned} \tag{60}$$

for the n_p position coordinates $p(t)$, the n_p velocity coordinates $v(t)$ and the n_λ Lagrange multiplier functions $\lambda(t)$, $n_\lambda < n_p$. $M(p,t) \in \mathbf{R}^{n_p, n_p}$ stands for the symmetric positive definite mass matrix, the vector $f(p,v,t) \in \mathbf{R}^{n_\lambda}$ denotes the applied forces and $G(p,t) := \frac{\partial}{\partial p} g(p,t)$ is the Jacobian of the holonomic constraints $g(p,t) \in \mathbf{R}^{n_\lambda}$. The system (60) has strangeness index 2.

The constraint $g(p,t)$ can be differentiated with respect to the time t, which yields the velocity constraint

$$0 = G(t,p)v + g^I(t,p), \tag{61}$$

where $g^I(t,p) = \frac{\partial}{\partial t} g(p,t)$.

The Euler–Lagrange equation (60) with the additional constraint (61) is an overdetermined DAE with strangeness index 1. The codes MEXX of Lubich/Nowak/Pöle/Engstler [27] and ODASSL of Führer [10] are constructed to solve this special type of overdetermined DAE's.

For this example we chose from [2]

$$\begin{gathered} g(p,t) = C(t)p(t) + r(t), \quad G(t) = C(t), \quad C(t) = \left[\begin{array}{cc} \sin(\nu t) & \cos(\nu t) \end{array} \right], \\ f(p,v,t) = -\nu^2 v(t) + q(t), \quad M(p,t) = I_2 \\ q(t) = \frac{e^t}{2-t} \left[\begin{array}{c} \sin(\nu t) + (2-t)(1+\nu^2) \\ \cos(\nu t) + (2-t)(1+\nu^2) \end{array} \right], \quad r(t) = -e^t(\sin(\nu t) + \cos(\nu t)) \end{gathered}$$

for $0 \le t \le 1$. Setting $x = [p^T, v^T, \lambda]^T$ we get a linear DAE ($n_p = 2, n_\lambda = 1$) with

$$E(t) = \left[\begin{array}{ccc} I_2 & 0 & 0 \\ 0 & I_2 & 0 \\ 0 & 0 & 0 \end{array} \right], \; A(t) = \left[\begin{array}{ccc} 0 & I_2 & 0 \\ 0 & -\nu^2 I_2 & -C^T(t) \\ C(t) & 0 & 0 \end{array} \right], \; f(t) = \left[\begin{array}{c} 0 \\ q(t) \\ r(t) \end{array} \right].$$

Choosing the initial conditions $x(0) = [1,1,1,1,0.5]^T$ the solution is $x(t) = [e^t, e^t, e^t, e^t, e^t/(2-t)]^T$. The output points are $t_{\text{out}} = 0.1, 0.2, \ldots, 1$.

EXAMPLE 4. This example is again a multibody system of the form (60). It is the linearized model of a 2–D truck introduced in [31]. This linear constant coefficient DAE has 23 unknowns, i.e. 11 position coordinates, 11 velocity coordinates and one Lagrange multiplier function.

Again, the exact solution of this problem is not known and we computed a reference solution at the output points $t_{\text{out}} = 2, 4, \ldots, 20$ in quadruple precision using tight tolerances.

EXAMPLE 5. Here we consider a problem which is not uniquely solvable. From [19] we take the differential–algebraic Riccati equation

$$\begin{aligned} E(t)^T \dot{X}(t) E(t) \quad &= \quad E(t)^T X(t) A(t) + A(t)^T X(t) E(t) + Q(t) \\ &\qquad -E(t)^T X(t) W(t) X(t) E(t), \end{aligned} \tag{62}$$

where

$$\begin{aligned} E(t) &= \left[\begin{array}{cc} 1-t^2 & -t+t^3 \\ 2t & -2t^2 \end{array} \right], \qquad W(t) = \left[\begin{array}{cc} 0 & 0 \\ 0 & 0 \end{array} \right], \\ A(t) &= \left[\begin{array}{cc} 3t - t^3 & t^4 - t^2 + 2t \\ 2t^2 + t^3 - t & -2t^3 + t^2 - 1 \end{array} \right], \\ Q(t) &= e^{-t} \left[\begin{array}{cc} Q_{11}(t) & Q_{12}(t) \\ Q_{21}(t) & Q_{22}(t) \end{array} \right] \text{ with} \\ & \quad Q_{11}(t) = -4t^2 - 4t(2t^2 + t^3 - t) \\ & \quad Q_{12}(t) = 4t^3 - 2t(-2t^3 + t^2 - 1) + 2t^2(2t^2 + t^3 - t) \\ & \quad Q_{21}(t) = Q_{12}(t) \\ & \quad Q_{22}(t) = -4t^4 + 4t^2(-2t^3 + t^2 - 1). \end{aligned}$$

Vectorizing equation (62) by going columnwise through $X(t)$ we get the linear DAE

$$\tilde{E}(t)\dot{x}(t) = \tilde{A}x(t) + f(t), \tag{63}$$

where

$$x(t) = \mathrm{vec}(X(t))$$
$$\tilde{E}(t) = E^T \otimes E^T$$
$$\tilde{A}(t) = A^T \otimes E^T + E^T \otimes A^T$$
$$f(t) = \mathrm{vec}(Q(t)).$$

Here we have used that $\mathrm{vec}(E^T X A) = (A^T \otimes E^T)\,\mathrm{vec}(X)$.

One solution of (62) is

$$X(t) = \begin{bmatrix} 0 & 0 \\ 0 & \exp(-t) \end{bmatrix}, \tag{64}$$

which is used to compute initial conditions at $t = 1$. Furthermore, the nonuniqueness in $X(t)$ is in kernel$(E(t))$ and therefore, $X(t)E(t)E(t)^+$ is unique (see [28]). Since the codes compute different solutions, we use the uniqueness of $X(t)E(t)E(t)^+$ to measure the error at $t_{\mathrm{out}} = 0.9, 0.8, \ldots, 0$.

EXAMPLE 6. Finally, we took the DAE

$$\begin{bmatrix} 0 & t \\ 0 & 0 \end{bmatrix} \dot{x}(t) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} \sin(t) \\ t \end{bmatrix}, \tag{65}$$

with solution $x(t) = \begin{bmatrix} -t & -t - \sin(t) \end{bmatrix}^T$. In $[-1, 1]$ our method is not applicable, since the matrix $E(t)$ has a rank change at $t = 0$. Locally the strangeness index jumps from one to zero in this point.

Example	MCONST	n	μ	d_μ	a_μ	u_μ
1.	yes	225	0	225	0	0
2.	yes	11	2	5	6	0
3.	no	5	2	2	3	0
4.	yes	23	2	20	3	0
5.	no	4	0	1	2	1
6. ($t \neq 0$)	no	2	1	0	2	0
6. ($t = 0$)	no	2	0	0	2	0

TABLE 1
Characteristic values of examples 1 to 6.

Table 1 gives an overview over the properties of the examples, we list

- MCONST, the statement if the matrices $\hat{E}$, $\hat{A}$ are constant
- n, the dimension of the problem
- μ, the strangeness index
- d_μ, a_μ, u_μ, the numbers of differential, algebraic and undetermined components, respectively.

We have tested the first five examples for different tolerances

$$\mathrm{ATOL} = \mathrm{RTOL} = 10^{-3+m/4}, \qquad m = 0, 1, 2, \cdots, 32.$$

For the examples 1 and 3, we know the exact solutions so that we were able to compute the error exactly by comparing analytic and computed results, whereas for Example 2 and Example 4 we used the computed reference solutions to determine the error. The error we compute is

$$\mathrm{ERR}(x) := \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - x_i^{\mathrm{ref}}}{|x_i| + 1}\right)^2}$$

taken over all output points, where x^{ref} is the exact or computed reference solution. For Example 5 x and x^{ref} are replaced by $\mathrm{vec}(XEE^+)$ and $\mathrm{vec}(X^{\mathrm{ref}}EE^+)$, respectively.

In order to investigate the efficiency of our code, we compared the error with the computing time. The resulting data are displayed in Fig. 1, 2 and 3 using logarithmic scales for both the X– and Y–axes.

We can generally state that our new methods can solve all given problems no matter what the index is. Whereas DASSL always fails for the higher index problem Example 3 and fails several times for Example 4, our methods work well for the first four examples. Furthermore, RADAU5 can treat the mechanical multibody systems of Example 3 and Example 4 only by suiting the tolerances to the corresponding index one, two and three variables, see [16]. Whereas RADAU5 underestimates the error for Example 3 and fails several times, RADAU5 overestimates the error for Example 4 for small torlerances and always fails if ATOL = RTOL $< 10^{-7}$. For Example 2 RADAU5 is not applicable due to the wrong order of the variables.

Additionally, we used MEXX and ODASSL for solving the multibody systems of Example 3 and Example 4. Both codes outperform the other codes for Example 3 and are much faster. For the 2–D truck (Example 4) our codes are competetive with ODASSL, whereas MEXX seems to converge against a different solution.

The Runge–Kutta method generally yields approximations which are of a higher precision than the ones obtained by the BDF method, because the Runge–Kutta method often overestimates the error. This becomes most obvious in Example 3 for low tolerances. The behaviour is due to the small integration intervals of size 0.1, which represents the maximum stepsize. Therefore, the solver always falls below the optimal stepsize, which increases the accuracy. The precision of the BDF method has in most cases the same order as the prescribed tolerances.

For a given tolerance the BDF code runs often faster than the Runge–Kutta code, especially for larger problems, but the number of function evaluations and the number of matrix factorizations are of the same order. This is in accordance with the theory. Note that for problems with time–varying coefficients the amount of work the Runge–Kutta solver has to do is mainly caused by the decomposition of the $(3n \times 3n)$–system, see Section 3.4.

The situation is different for Example 5, the Riccati equation with undetermined components. As before, for a given tolerance the Runge–Kutta method yields high precision approximations. The BDF method again achieves only approximations which are considerably less accurate and it always fails for tolerances smaller than 10^{-10}. Furthermore, the BDF method is much slower and needs more function evaluations. In the current implementation the BDF method is not competitive for DAE's with undetermined components. Note, that classical codes as DASSL and RADAU5 are not applicable for this example due to the undetermined components.

An exceptional case with respect to the analysis of the results is Example 6. We did not consider aspects of effectiveness, but concentrated on the fact that an index change happens at 0.0. As long as our methods do not hit exactly on 0.0 our codes do not realize the index change. As shown in Table 1, the index jumps at 0.0, but the numbers of the three characteristic values d_μ, a_μ, u_μ are constant. Therefore, this index jump does not cause any problems. Since the analysis of systems where the index or the other characteristic values have discontinuities is not completely settled yet, our codes always return the control with an error flag if a change in the characteristic values is detected.

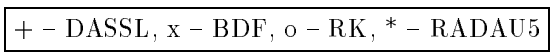

FIG. 1. *Comparison between the codes for Example 1 and 2*

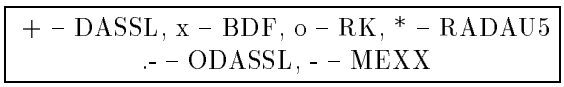

FIG. 2. *Comparison between the codes for Example 3 and 4*

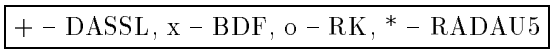

FIG. 3. *Comparison between the codes for Example 5*

6. Conclusion. In this paper we have described a new software package for the solution of linear differential–algebraic equations with variable coefficients. In comparision with other codes essentially no a priori information about the index and other characteristic values are necessary. The new code is able to treat problems with nonunique solutions and inconsistent initial conditions in the least squares sense.

A comparision with the available codes DASSL, RADAU5, ODASSL and MEXX shows that the new code is competitive, but applicable to a larger class of linear problems.

Currently an extended version of GELDA for nonlinear problems is under investigation.

REFERENCES

[1] E. ANDERSON, Z. BAI, C. BISCHOF, J. DEMMEL, J. DONGARRA, J. D. CROZ, A. GREENBAUM, S. HAMMERLING, A. MCKENNEY, S. OSTROUCHOV, AND D. SORENSON, *LAPACK Users' Guide*, Philadelphia, PA, 1992.

[2] U. M. ASCHER AND L. R. PETZOLD, *Stability of computation for constrained dynamics systems*, SIAM J. Sci. Statist. Comput., 14 (1993), pp. 95–120.

[3] K. E. BRENAN, S. L. CAMPBELL, AND L. R. PETZOLD, *Numerical Solution of Initial-Value Problems in Differential Algebraic Equations*, Elsevier, North Holland, New York, N. Y., 1989.

[4] S. L. Campbell, *Comment on controlling generalized state-space (descriptor) systems*, Internat. J. Control, 46 (1987), pp. 2229–2230.

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

[6] ———, *Nonregular descriptor systems with delays*, in Proc. Symp. Implicit & Nonlinear Systems, Dallas, 1992, pp. 275–281.

[7] S. L. Campbell and C. W. Gear, *The index of general nonlinear DAEs*, tech. report, North Carolina State University, Department of Mathematics, Raleigh, NC 27695-8205, 1993.

[8] P. Deuflhard, E. Hairer, and J. Zugck, *One step and extrapolation methods for differential-algebraic systems*, Numer. Math., 51 (1987), pp. 501–516.

[9] J. J. Dongarra, J. Du Croz, S. Hammerling, and R. J. Hanson, *Algorithm 656: An extended set of FORTRAN basic linear algebra subprograms*, ACMTM, 14 (1988), pp. 18–32.

[10] C. Führer, *Differential-Algebraische Gleichungsysteme in Mechanischen Mehrkörpersystemen*, PhD thesis, Math. Institute, Technische Universität, München, 1988.

[11] F. R. Gantmacher, *The Theory of Matrices*, vol. I, Chelsea Publishing Compan, New York, N.Y., 1959.

[12] C. W. Gear, *Differential-algebraic equation index transformations*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 39–47.

[13] C. W. Gear and L. R. Petzold, *Differential/algebraic systems and matrix pencils*, in Matrix Pencils, B. Kågström and A. Ruhe, eds., vol. 973 of Lecture Notes in Mathematics, Springer-Verlag, 1983, pp. 75–89.

[14] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, Maryland, 2nd ed., 1989.

[15] E. Hairer, C. Lubich, and M. Roche, *The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods*, Lecture Notes in Mathematics No. 1409, Springer-Verlag, Berlin, 1989.

[16] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II*, Springer-Verlag, Berlin, 1991.

[17] M. Hou, *Beobachtbarkeit und Beobachter von linearen Deskriptorsystemen: Theorie und Anwendung*. Proceedings of the Workshop 'Identifizierungs-, Analyse- und Entwurfsmethoden für mechanische Mehrkörpersysteme in Deskriptorform', Paderborn, 02.28 — 03.04.1994.

[18] ———, *A three–link planar manipulator model.* Sicherheitstechnische Reglungs- und Meßtechnik, Bergische Universität–GH Wuppertal, FRG, May 1994.

[19] P. Kunkel and V. Mehrmann, *Numerical solution of differential algebraic Riccati equations*, Linear Algebra Appl., 137/138 (1990), pp. 39–66.

[20] ———, *Smooth factorizations of matrix valued functions and their derivatives*, Numer. Math., 60 (1991), pp. 115–132.

[21] ———, *A new class of discretization methods for the solution of linear differential-algebraic equations.* Materialien LXII , FSP Mathematisierung, Universität Bielefeld, 1992. To appear in SIAM J. Numer. Anal.

[22] ———, *Canonical forms for linear differential-algebraic equations with variable coefficients*, J. Comput. Appl. Math., 56 (1994), pp. 225–259.

[23] ———, *A new look at pencils of matrix valued functions*, Linear Algebra Appl., 212/213 (1994), pp. 215–248.

[24] ———, *Local and global invariants of linear differential-algebraic equations and their relation*, Preprint SPC 95_25, TU Chemnitz-Zwickau, July 1995.

[25] P. Kunkel, V. Mehrmann, W. Rath, and J. Weickert, *GELDA: A software package for the solution of general linear differential algebraic equations*, Preprint SPC 95_8, TU Chemnitz-Zwickau, Feb. 1995.

[26] C. L. Lawson, R. J. Hanson, D. Kincaid, and F. T. Krogh, *Basic linear algebra subprograms for FORTRAN usage*, ACM Trans. Math. Software, 5 (1979), pp. 308–323.

[27] C. Lubich, U. Nowak, U. Pöle, and C. Engstler, *MEXX–Numerical software for the integration of constrained mechanical multibody systems*, Tech. Report SC 92-12, ZIB, Berlin, 1992.

[28] V. Mehrmann, *The Autonomous Linear Quadratic Control Problem*, Springer-Verlag, Berlin, 1991.

[29] L. R. Petzold, *A description of DASSL: A differential/algebraic system solver*, in IMACS Trans. Scientific Computing Vol. 1, R. S. Stepleman et al., eds., North-Holland, Amsterdam, 1983, pp. 65–68.

[30] P. J. Rabier and W. C. Rheinboldt, *Classical and generalized solutions of time-dependent linear differential algebraic equations*, Tech. Report ICMA-93-183, Dept. of Mathematics

and Statistics, Univ. of Pittsburgh, Pittsburgh, PA 15260, USA, 1993. To appear in Lin. Alg. Appl.

[31] B. Simeon, F. Grupp, C. Führer, and P. Rentrop, *A nonlinear truck model and its treatment as a multibody system*, J. Comput. Appl. Math., 50 (1994), pp. 523–532.