InitDAE: Computation of consistent values, index determination and diagnosis of singularities of DAEs using automatic differentiation in Python

Diana Estévez Schwarz
Beuth Hochschule für Technik Berlin[*]

René Lamour
Humboldt Universität zu Berlin, Institut für Mathematik [†]

April 5, 2019

Abstract

InitDAE is a prototype written in Python that computes consistent initial values of differential-algebraic equations (DAE), determines their index and a related condition number that permits the diagnosis of singularities. The algorithm for the consistent initialization uses a projector based constrained optimization approach and the inherent differentiations are provided by automatic differentiation (AD), using AlgoPy. Consequently, a detailed description of the local structural properties of the DAE becomes possible using the SVD. InitDAE has been conceived for academic purposes and is well-suited for examples of moderate size. In this article we give an overview of the algorithm, show actual features and discuss future possibilities, in particular the integration with Taylor series methods.

Keywords: DAE, differential-algebraic equation, consistent initial value, integration, index, derivative array, projector based analysis, nonlinear constrained optimization, SQP, automatic differentiation

MSC-Classification: 65L05, 65L80, 34A09, 34A34, 65D25, 90C30, 90C55

1 Introduction

Conventional integration methods cannot cope with general, higher-index DAEs. Hence, the diagnosis of the properties and, particularly, the structure of DAEs is crucial to guarantee the reliability of numerical results. While for some classes of DAEs the structure is well understood (e.g. DAEs in Hessenberg form and DAEs resulting in circuit simulation), a detailed analysis of the obtained DAEs becomes necessary if new types of elements are included or different classes of equations are coupled. In this article we give an overview of the implementation of InitDAE, a software package developed for this purposes, cf. [1].

[*]estevez@beuth-hochschule.de
[†]lamour@math.hu-berlin.de

Our starting point will be a projector based decoupling of components into the differentiated and the undifferentiated part. To this end, we consider

$$f(x', x, t) = 0, \quad \text{where } f_{x'} \text{ is singular,} \tag{1}$$

and assume that

$$\ker f_{x'}(x', x, t)$$

does not depend on (x', x) and that a continuously differentiable projector $Q(t)$ onto $\ker f_{x'}$ exists. On the basis of the complementary projector $P(t) := I - Q(t)$ we can then reformulate the DAE as

$$f(x', x, t) = f(Px', x, t) = f((Px)' - P'x, x, t) = 0, \tag{2}$$

as already introduced in [2]. In this sense, we will use the notation:

- Px for the differentiated component,
- Qx for the undifferentiated component,

since, for the decoupling $x' = (Px)' + (Qx)'$, we can see that $(Px)' = \varphi_1(x, t)$ is implicitly given, cf. [3], [4].

Recall further that the singularity of $f_{x'}$ means that (1) contains derivative-free equations, called constraints, and that the differentiation of (1) may lead to further derivative-free equations, called hidden constraints.

According to [4], for an initial guess $\alpha \in \mathbb{R}^n$, consistent initial values x_0 can be computed solving the following constraint optimization problem

$$\min \quad \|P(x_0 - \alpha)\|_2 \tag{3}$$

$$\text{subject to all explicit and hidden constraints.} \tag{4}$$

In this paper, on the one hand, we present how this approach is being implemented. To this end, in Section 2, we start describing how the solvability of (3) - (4) is related to the index of the DAE. Since, on the other hand, we also want to present some new aspects not yet considered in [4], in Section 3 we introduce a more general initialization algorithm that involves additional user-given requirements. In Section 4 we summarize the main aspects within the theoretical framework of InitDAE before we focus on the implementation using automatic differentiation in Section 5.

InitDAE opens new possibilities for the integration with Taylor series methods and for monitoring singularities, which are described briefly in the Sections 6 and 7, respectively. We finally illustrate the obtained results for some well-known examples from literature in Section 8.

In the Appendix we give an overview of some properties of computations with automatic differentiation in order to be comprehensible to readers without expertise in this field and to motivate our notation.

2 Reinterpretation of the differentiation index

With the decoupling $x = Px + Qx$ we can use the following definition of the differentiation index, which was introduced in [3], [4].

Definition 1 *The differentiation index is the smallest integer μ such that*

$$\begin{aligned} f(x',x,t) &= 0, \\ \frac{d}{dt} f(x',x,t) &= 0, \\ &\vdots \\ \frac{d^{\mu-1}}{dt^{\mu-1}} f(x',x,t) &= 0, \end{aligned}$$

uniquely determines Qx as a function of (Px,t).

Due to (2) there exists a function φ_1 such that locally

$$(Px)' = \varphi_1(x,t)$$

holds. If, according to Definition 1, there exists another function φ_2 such that

$$Qx = \varphi_2(Px,t),$$

then one further differentiation provides

$$(Qx)' = \varphi_3((Px)',Px,t) = \tilde{\varphi}_3(x,t).$$

Consequently, if μ is the differentiation index according to Definition 1, then the conventional differentiation index (see e.g. [5]) results to be μ as well.

In order to allow for the differentiations, we consider

$$F_j(x^{(j+1)}, x^{(j)}, \ldots, x', x, t) := \frac{d^j}{dt^j} f(x',x,t),$$

and define for $z_i \in \mathbb{R}^n$, $i = 0, \ldots, k$,

$$g^{[k+1]}(z_0, z_1, \ldots, z_{k+1}, t) := \begin{pmatrix} f(z_1, z_0, t) \\ F_1(z_2, z_1, z_0, t) \\ \vdots \\ F_k(z_{k+1}, \ldots, z_0, t) \end{pmatrix}. \tag{5}$$

With this notation, we can formulate the constraint optimization problem (3)-(4) more precisely

$$\min \quad \|P(z_0 - \alpha)\|_2 \tag{6}$$

$$\text{subject to} \quad g^{[k+1]}(z_0, z_1, \ldots, z_{k+1}, t) = 0, \quad k \geq \mu - 1 \tag{7}$$

for $\mu \geq 1$. We will assert in Section 3 that the amount of consistent values $(z_0, z_1, \ldots, z_m)$ for $(x(t_0), x^{(1)}(t_0), \ldots, x^{(m)}(t_0))$, with $m \leq k$ depends on k and on the index μ, and it consequently has to be determined.

To compute the index μ in this context, for $z_i \in \mathbb{R}^n$, $j = 0, \ldots, k$, we denote by

$$G^{[k]}_{(z_0)}(z_0, z_1, \ldots, z_k, t) \in \mathbb{R}^{nk \times n}$$

the Jacobian matrix of $g^{[k]}(z_0, z_1, \ldots, z_k, t)$ with respect to z_0, by

$$G^{[k]}_{(z_1,\ldots,z_k)}(z_0, z_1, \ldots, z_k, t) \in \mathbb{R}^{nk\times nk}$$

the Jacobian matrix of $g^{[k]}(z_0, z_1, \ldots, z_k, t)$ with respect to $(z_1, \ldots, z_k)$ and consider the matrices

$$\mathcal{B}^{[k]} := \begin{pmatrix} P & 0 \\ \\ G^{[k]}_{(z_0)} & G^{[k]}_{(z_1,\ldots,z_k)} \end{pmatrix} \in \mathbb{R}^{n(k+1)\times n(k+1)}, \quad k = 1, \ldots.$$

According to [4], we check if the matrices $\mathcal{B}^{[k]}$ are 1-full with respect to the first n columns for $k = 1, 2, \ldots$, i.e., whether

$$\ker \mathcal{B}^{[k]} \subseteq \left\{ \begin{pmatrix} s_0 \\ s_1 \end{pmatrix} \;:\; s_0 \in \mathbb{R}^n,\; s_0 = 0,\; s_1 \in \mathbb{R}^{nk} \right\}. \tag{8}$$

InitDAE concludes that the index is μ if μ is the smallest integer for which $\mathcal{B}^{[\mu]}$ is one-full.

3 Initialization without/with additional requirements

In [4] we further analyzed how the solvability of the optimization problem (6) - (7) depends on the 1-fullness of the matrix $\mathcal{B}^{[\mu]}$ under the assumption that

$$G^{[\mu]} := \begin{pmatrix} G^{[\mu]}_{(z_0)} & G^{[k]}_{(z_1,\ldots,z_\mu)} \end{pmatrix}$$

has full row rank. According to [6], this minimal solution could also be computed setting

$$\Pi(z_0 - \alpha) = 0$$

for a suitable orthogonal projector Π with rank $\Pi = d$, where d is the degree of freedom. In fact, if we denote by W_1 and W_2 orthogonal projectors fulfilling

$$\ker W_1 = \operatorname{im} G^{[\mu]}_{(z_1,\ldots,z_\mu)},$$
$$\ker W_2 = \operatorname{im} W_1 G^{[\mu]}_{(z_0)} Q,$$

then Π is the orthogonal projector with

$$\operatorname{im} \Pi = \ker \begin{pmatrix} Q \\ W_2 W_1 G^{[\mu]}_{(z_0)} \end{pmatrix}.$$

The computation of consistent initial values solving the optimization problem (6) - (7) precisely assigns $\Pi x = \Pi\alpha$, cf. [6].

If the user wants to prescribe initial values to selected components or special relations between initial values, this can be done by defining additional restrictions. In that case, we have to check whether these additional equations are admissible. Let us denote by

$$u(z_0) = 0$$

these user-given restrictions for the initial values. If these equations are independent of $g^{[\mu]}$ or, more precisely, if

$$G_u^{[\mu]} := \begin{pmatrix} G^{[\mu]}_{(z_0)} & G^{[\mu]}_{(z_1,\ldots,z_\mu)} \\ U_{(z_0)} & 0 \end{pmatrix}$$

has full row rank for the Jacobian $U_{(z_0)}$ of u with respect to z_0, then our initialization procedure can easily be adapted, leading to

$$\min \quad \|P(z_0-\alpha)\|_2 \tag{9}$$
$$\text{subject to} \quad g^{[k+1]}(z_0,z_1,\ldots,z_{k+1},t)=0, \quad u(z_0)=0 \tag{10}$$

for $k \geq \mu - 1$. Since u is supposed to hold only at the initial point t_0, no derivatives of u are involved. The corresponding projector-based description would be

$$\Pi_u(x-\alpha)=0$$

with rank $\Pi_u = d - \text{rank}\,(U_{(z_0)})$. Further, this projector Π_u is the orthogonal projector fulfilling

$$\text{im}\,\Pi_u = \ker \begin{pmatrix} Q \\ W_2W_1 \begin{pmatrix} G^{[\mu]}_{(z_0)} \\ U_{(z_0)} \end{pmatrix} \end{pmatrix},$$

where the orthogonal projectors W_1 and W_2 are now defined by

$$\ker W_1 = \text{im} \begin{pmatrix} G^{[\mu]}_{(z_1,\ldots,z_\mu)} \\ 0 \end{pmatrix},$$
$$\ker W_2 = \text{im}\, W_1 \begin{pmatrix} G^{[\mu]}_{(z_0)} \\ U_{(z_0)} \end{pmatrix} Q,$$

In both cases, if the approach is performed with $k > \mu$, we will not only obtain consistent initial values $z_0 = x(t_0)$, but also consistent values for the derivatives

$$z_1,\ldots,z_{k-\mu}.$$

4 The setting of InitDAE

In order to implement the approach described above, some framework conditions have to be taken into account.

4.1 Main mathematical aspects

We have chosen some specific numerical approaches to cope with the different tasks:

- Differentiation:
  Set up of the derivative array (5) and the related Jacobian matrices. We use automatic differentiation (AlgoPy, cf. [7]).

- Linear Algebra:
  Determination of rank and condition number, bases for nullspaces and projectors. We use the singular value decomposition and the resulting criteria for 1-fullness presented in [3] and [4].

- Optimization:
  Constrained optimization problem to compute initial values and Taylor coefficients. We use sequential least squares programming (SLSQP), which was also analyzed in [4].

4.2 Formulations of the DAE

DAEs can be formulated in different forms. In InitDAE, three different formulations of the DAE are implemented via the options

- linear: f is given as $f(x', x, t) = A(t)x' + B(t)x - q(t)$ with given matrices A, B and a vector q.
- standard: f is given for $f(x', x, t)$,
- proper: $f_p(d_1, x_0, t)$ and $d(x,t)$ are given for $f_p(d'(x,t), x, t)$.

5 How InitDAE Works

In the following, we focus on the realization with automatic differentiation, which was not addressed in [4]. Hence, we describe how we set up and solve the optimization problem (6)–(7) or, if additional requirements are used, (9)–(10).

5.1 Setting up the optimization problem with automatic differentiation

Since InitDAE uses automatic differentiation, we reformulate the original problem using truncated Taylor series with $D = K + 1$ coefficients

$$x^{[0:K]} = [c_0, c_1, c_2, \dots, c_K] = \left[z_0, z_1, \frac{z_2}{2}, \dots, \frac{z_K}{K!}\right],$$

for the unknown function $x(t)$ at a time-point t_0 that is given[1]. For $k < K = D - 1$ we use the relationship

$$\begin{aligned} x(t^{[0:k+1]}) &= [c_0, c_1, c_2, \dots, c_{k+1}] \\ x'(t^{[0:k]}) &= [c_1, 2c_2, \dots, (k+1)c_{k+1}] \end{aligned}$$

and reformulate our constraints.

- In the standard (and linear) case we consider the function f

$$\begin{aligned} f^{[0:k]} &= f\left([c_1, 2c_2, \dots, (k+1)c_{k+1}], [c_0, c_1, c_2, \dots, c_k], t^{[0:k]}\right) \\ &=: \left[f^{[0]},\ f^{[1]}, \dots, f^{[k]}\right]. \end{aligned}$$

[1]The notation is illustrated in the Appendix.

- In the proper case, in a first step, we consider

$$d^{[0:k]} = d([c_0, c_1, c_2, \ldots, c_k], t^{[0:k]}) = \left[d^{[0]}, \ldots, d^{[k]}\right]$$

and, correspondingly,

$$d'([c_0, c_1, c_2, \ldots, c_{k+1}], t^{[0:k]}) = \left[d^{[1]}, 2d^{[1]}, \ldots, (k+1)!\ d^{[k+1]}\right]$$

and set

$$\begin{aligned} f^{[0:k]} &= f_p\left(\left[d^{[1]}, 2d^{[1]}, \ldots, (k+1)!\ d^{[k+1]}\right], [c_0, c_1, c_2, \ldots, c_k]\right), t^{[0,k]}) \\ &=: \left[f^{[0]},\ f^{[1]}, \ldots, f^{[k]}\right]. \end{aligned}$$

With this notation, we obtain for a fixed $t^{[0,k]}$ the constraints

$$constraints(c_0, c_1, c_2, \ldots, c_{k+1}) := \begin{pmatrix} f^{[0]}(c_0, c_1) \\ f^{[1]}(c_0, c_1, c_2) \\ f^{[2]}(c_0, c_1, c_2, c_3) \\ \vdots \\ f^{[k]}(c_0, c_1, \ldots, c_{k+1}) \\ u(c_0) \end{pmatrix}$$

and the objective function $\|P(c_0 - \alpha)\|_2$.

To solve the optimization problem we use the method SLSQP of the module scipy.optimize.minimize from the SciPy library. This code needs the objective function, the constraints and their Jacobian matrices.

- The Jacobian matrix of the objective function, which is a gradient in this case, reads
$$\frac{1}{\|P(c_0 - \alpha)\|_2}(P(c_0 - \alpha))^T.$$

- The Jacobian matrix of the constraints will be provided considering the Taylor coefficients of the following matrix functions
  - in the cases *linear* and *standard*
$$A(x', x, t) = \frac{\partial f}{\partial x'} \quad \text{and} \quad B(x', x, t) = \frac{\partial f}{\partial x}$$
  - in the case *proper*
$$A(x', x, t) = \frac{\partial f_p}{\partial d_1} \cdot \frac{\partial d}{\partial x} \quad \text{and} \quad B(x', x, t) = \frac{\partial f_p}{\partial x_0} + \frac{\partial f_p}{\partial d_1} \cdot \frac{d}{dt}\frac{\partial d}{\partial x}$$

    for $f_p(d_1, x_0, t)$.

The Jacobian of the objective function can be derived straightforward. In contrast, to provide the Jacobian of the constraints we assemble information on the computation with AD. Let us therefore consider

$$\begin{aligned}
A^{[0:k]} &:= A([c_1, 2c_2, \ldots, kc_{k+1}], [c_0, c_1, c_2, \ldots, c_k], t^{[0:k]}) \\
&= \left(A^{[0]}, \quad \ldots, \quad A^{[k]}\right) \in \mathbb{R}^{m\times m\times(k+1)}, \\
B^{[0:k]} &:= B([c_1, 2c_2, \ldots, kc_{k+1}], [c_0, c_1, c_2, \ldots, c_k], t^{[0:k]}) \\
&= \left(B^{[0]}, \quad \ldots, \quad B^{[k]}\right) \in \mathbb{R}^{m\times m\times(k+1)},
\end{aligned}$$

and, in case that additional requirements u are given at t_0,

$$U(x,t) = \frac{\partial u}{\partial x}$$

and

$$U^{[0]} := U(c_0, t_0)$$

Using this notation, the Jacobian matrix of the constraints is given by

$$\begin{pmatrix}
B^{[0]} & A^{[0]} & & & & \\
B^{[1]} & B^{[0]} + A^{[1]} & 2A^{[0]} & & & \\
B^{[2]} & B^{[1]} + A^{[2]} & B^{[0]} + 2A^{[1]} & 3A^{[0]} & & \\
B^{[3]} & B^{[2]} + A^{[3]} & B^{[1]} + 2A^{[2]} & B^{[0]} + 3A^{[1]} & 4A^{[0]} & \\
\vdots & & & \ddots & \ddots & \\
B^{[k]} & \cdots & \cdots & \cdots & \cdots & (k+1)A^{[0]} \\
U^{[0]} & 0 & \cdots & \cdots & 0 & 0
\end{pmatrix}.$$

For clarity, we illustrate this for the linear case, no additional requirements u and $k = 2$. In this case, the original derivative array reads

$$\begin{pmatrix} f(x', x, t) \\ \frac{d}{dt} f(x', x, t) \\ \frac{d^2}{dt^2} f(x', x, t) \end{pmatrix} = \begin{pmatrix} B & A & 0 & \\ B' & B + A' & A & \\ B'' & 2B' + A'' & B + 2A' & A \end{pmatrix} \begin{pmatrix} x \\ x' \\ x'' \\ x''' \end{pmatrix} - \begin{pmatrix} q \\ q' \\ q'' \end{pmatrix} = 0.$$

In terms of Taylor coefficients, this corresponds to

$$\begin{pmatrix} f^{[0]} \\ f^{[1]} \\ 2f^{[2]} \end{pmatrix} = \begin{pmatrix} B^{[0]} & A^{[0]} & 0 & \\ B^{[1]} & B^{[0]} + A^{[1]} & A^{[0]} & \\ 2B^{[2]} & 2B^{[1]} + 2A^{[2]} & B^{[0]} + 2A^{[1]} & A^{[0]} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ 2c_2 \\ 6c_3 \end{pmatrix} - \begin{pmatrix} q^{[0]} \\ q^{[1]} \\ 2q^{[2]} \end{pmatrix} = 0.$$

Rearranging the matrix multiplication and scaling the latter equation, we obtain

$$\begin{pmatrix} f^{[0]} \\ f^{[1]} \\ f^{[2]} \end{pmatrix} = \begin{pmatrix} B^{[0]} & A^{[0]} & 0 & \\ B^{[1]} & B^{[0]} + A^{[1]} & 2A^{[0]} & \\ B^{[2]} & B^{[1]} + A^{[2]} & B^{[0]} + 2A^{[1]} & 3A^{[0]} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} - \begin{pmatrix} q^{[0]} \\ q^{[1]} \\ q^{[2]} \end{pmatrix} = 0.$$

This clear block Hessenberg structure allows a straightforward implementation of the Jacobian matrix.

5.2 Realization with AlgoPy

For linear DAEs, the Taylor coefficients of the matrices $A(t)$ and $B(t)$ directly provide the entries for the Jacobian matrix. For nonlinear DAEs, the computation of the Taylor coefficients of the Jacobian matrices requires a tracing of the computational graph of f, which, in AlgoPy, can be realized using the module CGraph. As AlgoPy realizes the so-called *forward* and *reverse* mode, we can compute Taylor coefficients of A and B, too. Using the obtained graph, we assemble the Jacobian matrices depending on the representation of the DAE. The Taylor coefficients of f representing the derivative array are a byproduct of these computations.

6 Integration with Taylor Series Method

In the past, AD was applied to DAEs by Chang and Corliss [8], using the differentiation index by Campbell and Hollenbeck [9] and using the structural index by Nedialkov and Pryce [10],[11], and by Barrio [12]. The tractability index concept was applied to the index determination by Lamour and Monett Diaz [13], [14] and used for the integration of DAEs of index up to 2 in [15].

The above described computation of Taylor coefficients opens new possibilities for the integration with Taylor series methods that are based on the differentiation index from Definition 1 and can be formulated straightforward. For $j \geq 0$, we denote by $(c_\ell)_j$ the ℓ-th Taylor coefficients of x at the time-point t_j and α_j the initial guess for $(c_0)_j$. With this notation, our algorithm can be formulated at t_{j+1}, for $h_j = t_{j+1} - t_j$ as follows:

- Obtain a Taylor approximation for the initial guess

$$\alpha_{j+1} = \sum_{\ell=0}^{K-\mu} (c_\ell)_j h_j^\ell \quad (\approx x(t_j + h_j))$$

- Set starting points for the iteration

$$(c_i)_j = \sum_{\ell=0}^{K-\mu} d_{\ell,i} h_j^\ell \quad \left(\approx \frac{1}{i!} x^{(i)}(t_j + h_j)\right)$$

  using the conventional matrix product and i-th power

$$(d_{0,i}, \ldots, d_{K-\mu,i}) = ((c_0)_j, \ldots, (c_{K-\mu})_j) \cdot \begin{pmatrix} 0 & & & \\ 1 & \ddots & & \\ & \ddots & \ddots & \\ & & (K-\mu) & 0 \end{pmatrix}^i$$

  for $i = 0, \ldots, K - \mu$.

- Compute for t_{j+1} instead of t_0 and without u in the constraints (since these requirements are considered at t_0 only)

$$\begin{array}{rl} \min & \|P((c_0)_{j+1} - \alpha_{j+1})\|_2 \\ \text{subject to} & \textit{constraints}((c_0)_{j+1}, (c_1)_{j+1}, \ldots, (c_k)_{j+1}, (c_{k+1})_{j+1}) = 0. \end{array}$$

- Set
$$x_{j+1} = (c_0)_{j+1}.$$

Note that this is a completely different approach compared to [15], where the Taylor-Method was applied to the inherent ODE for DAEs of index up to 2. Here, instead, we do not have a restriction for the index, at least theoretically. The approach also differs considerably from the method using dummy derivatives, cf. [11] and the references therein, because the optimization approach in fact corresponds to a projection onto the solution manifold.

7 Diagnosis

In [3], we presented results concerning the diagnosis of singularities. These investigation were a crucial motivation for developing InitDAE. With the above Taylor integration method, we can finally monitor the singularities during the integration, analogously as in [16]. We emphasize that in [16] the tractability index matrix sequence from [17] was analyzed. Now, we compute a condition number and further indicators described in [3] at each time point. All required matrices can be provided as a byproduct during the integration.

As pointed out in [3], the condition number should not be interpreted in dependence of a particular scaling. It gives a hint to a singularity if its graphical representation suggests a pole, cf. Example 8.3.

8 Examples

In this section, we discuss some features of InitDAE considering three well-known examples from the DAE literature.

8.1 Pendulum

Testing the classical index-3-Pendulum with Modelica 1.12.0, we became aware of an unexpected effect. If we simulate

```
model DAEPendulum "DAE Pendulum"
  parameter Real g = 1;
  parameter Real m = 1;
  parameter Real l = 1;
  Real x(start = 0.5);
  Real y(start = 0.5);
  Real vx(start = 0);
  Real vy(start = 0);
  Real lambda(start = 0);
equation
  der(x) = vx;
  der(y) = vy;
  m * der(vx) = 2 * lambda * x;
  m * der(vy) = 2 * lambda * y - m * g;
  x ^ 2 + y ^ 2 = l ^ 2;
  annotation(experiment(StartTime = 0.0, StopTime = 12.0,
```

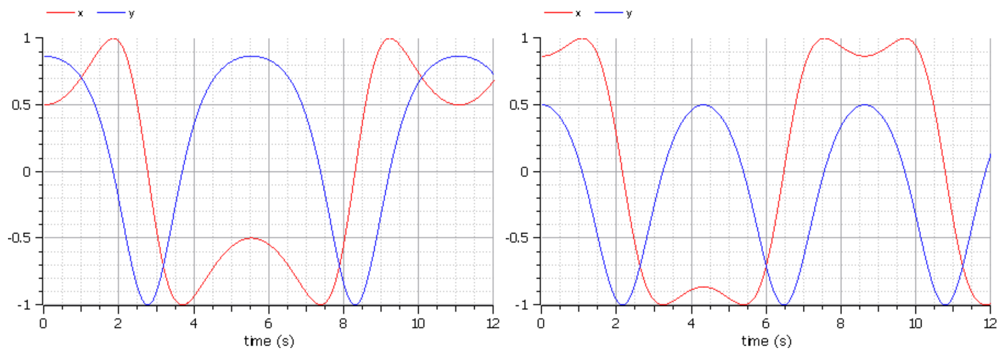

Figure 1: Results delivered by Modelica defining first v_x and then v_y (left) and vice versa (right).

```
                                                    Tolerance = 1e-006));
end DAEPendulum;
```

then we obtained the solution represented on Figure 1 (left). Consequently, the structural analysis leads to a preservation of the initial value for $x(0)$. $y(0)$ is computed according to the explicit constraints. In contrast, if, in the model, we define the variables in the order

```
(...)
  Real x(start = 0.5);
  Real y(start = 0.5);
  Real vy(start = 0);
  Real vx(start = 0);
  Real lambda(start = 0);
(...)
```

and leave the equations unchanged, then the initial value for $y(0)$ is preserved and $x(0)$ is computed accordingly, cf. Figure 1 (right). This effect seems to result from the graph theoretical background of the method, see [18].

Our method results from a projector based analysis and, for the position coordinates of the pendulum it computes the nearest point on the circle, i.e., for the initial guess $(0.5, 0.5)$ we obtain $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$, as can be seen in Figure 2 (left). The integration with initial guess $(0.5, 0.5, 0, 0, 0)$ over an interval $[0, 12]$ provides the components plotted in Figure 2 (left). The condition number computed during the integration and the one obtained along a previously computed solution are identical (see Figure 2 (right)). We do not observe here a singularity, but the condition number has its maximal value when the pendulum has its maximal horizontal speed, i.e., $x_1 = 0$.
For the pendulum example, we computed the projector Π as

$$\Pi = \begin{pmatrix} 0.5 & -0.5 & & & \\ -0.5 & 0.5 & & & \\ & & 0.5 & -0.5 & \\ & & -0.5 & 0.5 & \\ & & & & 0 \end{pmatrix}$$

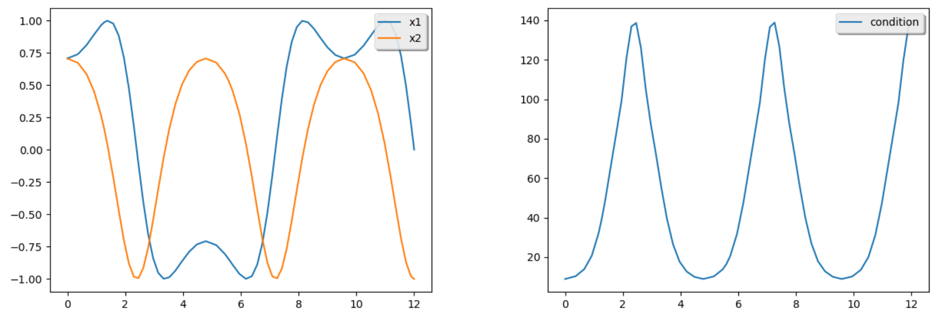

Figure 2: Solution components without additional conditions (left) and condition number (right) obtained with InitDAE

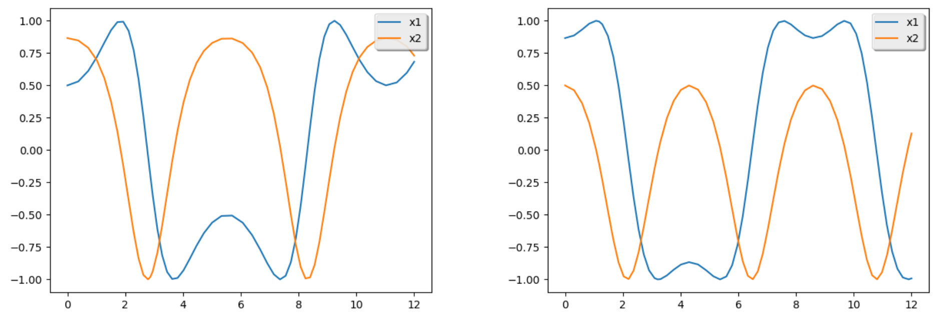

Figure 3: Solution of the pendulum with additional condition $x_1 = 0.5$ (left) or $x_2 = 0.5$ (right) obtained with InitDAE

for an initial guess α with $\alpha_1 = \alpha_2$ and $\alpha_3 = \alpha_4 = 0$. This means, on the other hand, that also the consistent initial value fulfills $x_1 = x_2$, as shown above. If we use additional conditions for the state variables x_1 or x_2, then, in the resulting projector Π_u the first block vanishes and, if we use additional conditions for the velocity variables x_3 or x_4, the second block vanishes.
If we want to fix the initial position of the pendulum at $x_1 = 0.5$, we use the additional function $u = x_1 - 0.5$. The integration result is plotted in Figure 3 (left). If we fix $x_2 = 0.5$ instead, we obtain the trajectory plotted in Figure 3 (right) analogously. These results are identical with those from Modelica (Figure 1), but they are the result of a conscious decision of the user and not delivered by chance, depending of the order of lines in the code. However, Modelica also offers options for a prioritization of the initial values.

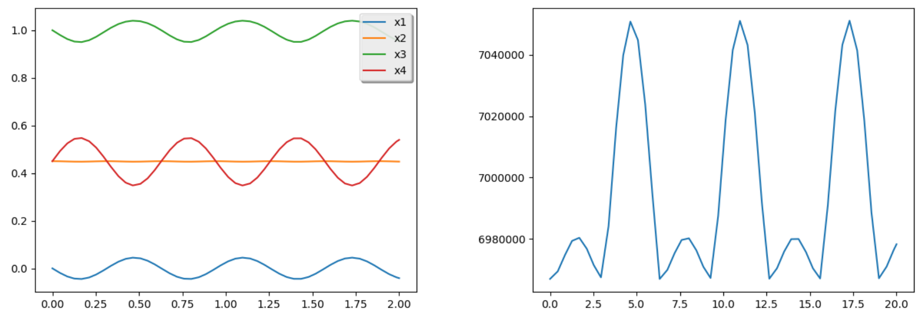

Figure 4: Solution and condition number of car axis example

8.2 Car Axis

Let us focus on the Car Axis problem described in [19]. The structure of the index-3 equations corresponds to

$$
\begin{aligned}
p' &= q \\
Kq' &= f(\omega \cdot t, p, \lambda), \quad p, q \in \mathbb{R}^4,\ \lambda \in \mathbb{R}^2, \quad 0 \le t \le 3, \\
0 &= \phi(\omega \cdot t, p),
\end{aligned}
$$

for $\omega = 10$. All the remaining parameters and notations used in the following correspond to [19]. In order to avoid a disadvantageous scaling of the Taylor coefficients, we change the independent variable t to $\tau = 10\ t$, obtaining the DAE

$$
\begin{aligned}
p' &= q/10 \\
Kq' &= f(\tau, p, \lambda)/10, \quad p, q \in \mathbb{R}^4, \quad 0 \le \tau \le 0.3, \\
0 &= \phi(\tau, p).
\end{aligned}
$$

For the initialization, we started with the inconsistent values

```
L   = 1e0          xra  = -L0/L*10
L0  = 0.5e0        xla  = xra
xr   = L           yra  = 0e0
xl   = 0           yla  = 0e0
yr   = L0          lam1 = 0e0
yl   = yr          lam2 = 0e0
```

Observe that the condition number is high, but does not suggest the existence of a singularity.
Note further that, before the solution was plotted in Figure 4, the independent variable was rescaled to its original value $t = \frac{\tau}{10}$.

8.3 Robotic Arm

Finally, we touch the index 5 Robotic Arm problem introduced in the known form by Campbell in [20], but better available, e.g., in Campbell, Griepentrog

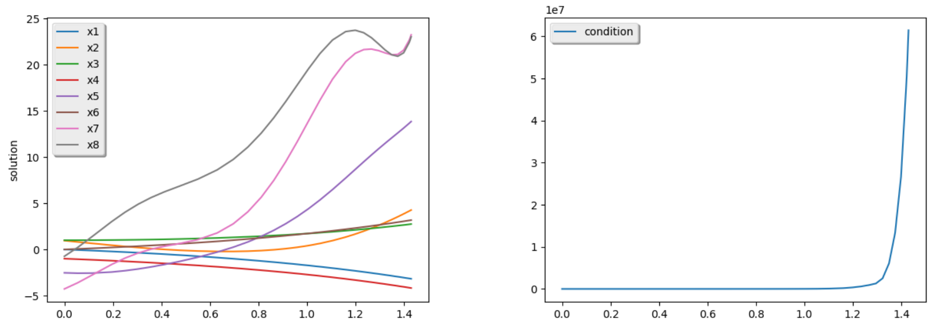

Figure 5: Solution and condition number of the robotic arm problem.

[21].
The integration over the interval [0,1.5] discovers a singularity at the end of the interval. Therefore, we plot the result and the condition number over the interval [0,1.43] in Figure 5. A detailed discussion will be published soon in [22].

9 Conclusion

InitDAE (see [1]) is the implementation of a bunch of algorithms developed by the authors in the last couple of years to initialize DAEs, to characterize their structure, to monitor singularities and to integrate them with Taylor series methods. It has been conceived for academic purposes and delivers precise information thanks to algorithmic differentiation. For examples of moderate size (we tested DAEs of dimension up to 600, cf. [23]), the run time for the consistent initialization was acceptable. In contrast the integration is recommendable for small examples only, due to the computational costs. Nevertheless, in our opinion, it is a valuable tool for diagnosis purposes.

A Taylor coefficients and Automatic Differentiation

For the details on automatic (or algorithmic) differentiation we refer to [24]. Here, we merely clarify our notation and the aspects we utilize.

A.1 Some Basics

Automatic differentiation works with truncated Taylor series, considering D coefficients. If the Taylor variable is initialized with t_0 for $K = D - 1$, we obtain

$$t^{[0:K]} := [t_0, \quad 1, \quad 0, \quad ,\dots, \quad 0]$$

and the Taylor expansion of $x(t^{[0:K]})$ corresponds to

$$\begin{aligned} x^{[0:K]} &:= \left[x(t_0), \quad x'(t_0), \quad \frac{x''(t_0)}{2!}, \quad \dots, \quad \frac{x^K(t_0)}{K!} \right] \\ &=: [c_0, c_1, \dots, c_K]. \end{aligned}$$

For a matrix function we proceed analogously (cf. [15]), i.e., for $B(x,t) \in \mathbb{R}^{m \times n}$ we will use the notation

$$B^{[0:K]} := B(x^{[0:K]}, t^{[0:K]}).$$

Example 1 • *For $D = 5$, $K = D - 1 = 4$, $t_0 = 1$:*

$$\begin{aligned} t^{[0:4]} &:= [1, \quad 1, \quad 0, \quad 0, \quad 0], \\ \exp(t^{[0:4]}) &= [2.7183, \quad 2.7183, \quad 1.3591, \quad 0.4530, \quad 0.1133], \end{aligned}$$

- *For $D = 3$, $K = 2$, $t^{[0:2]} := [\pi, \quad 1, \quad 0]$ and a matrix-valued function*

$$B(t) = \begin{pmatrix} \cos(t) & -\sin(t) \\ \sin(t) & \cos(t) \end{pmatrix}$$

we obtain

$$\begin{aligned} B(t^{[0:2]}) &= \begin{pmatrix} \cos(t^{[0:2]}) & -\sin(t^{[0:2]}) \\ \sin(t^{[0:2]}) & \cos(t^{[0:2]}) \end{pmatrix} \\ &= \left[\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}, \quad \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \quad \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \right] \\ &=: \left[B^{[0]}, \quad B^{[1]}, \quad B^{[2]} \right] := B^{[0:2]} \in \mathbb{R}^{2 \times 2 \times 3}. \end{aligned}$$

A.2 Product rule in AD

We consider the product

$$h(t) = u(t) \cdot v(t)$$

of two analytic functions u and v. Higher-order derivatives of h are determined by Leibniz's rule

$$h^{(k)}(t) = \sum_{i=0}^{k} \binom{k}{i} u^{(i)}(t) v^{(k-i)}(t).$$

As a consequence, the result of the product of two Taylor objects

$$h(t^{[0:K]}) = u(t^{[0:K]}) \cdot v(t^{[0:K]})$$

can be computed with

$$[h^{[k]}] = \sum_{i=0}^{k} u^{[i]} v^{[k-i]}, \quad k = 0, \dots, K. \tag{11}$$

Example 2 • *For $D = 5$, $K = D - 1 = 4$, $t_0 = 1$:*

$$t^{[0:4]} \exp(t^{[0:4]}) = [2.7183, \quad 5.4366, \quad 4.0774, \quad 1.8122, \quad 0.5663].$$

- *For $D = 3$, $K = D - 1 = 2$, $t_0 = \pi$ and B from Example 1 we compute the product*

$$q(t^{[0:2]}) = B(t^{[0:2]}) \cdot \begin{pmatrix} \sin\bigl(2\,t^{[0:2]}\bigr) \\ \cos\bigl(2\,t^{[0:2]}\bigr) \end{pmatrix}$$

  and obtain, using the product rule (11) *with the Taylor coefficients*

$$\sin(2t^{[0:2]}) = [0, 2, 0], \quad \cos(2t^{[0:2]}) = [1, 0, -2],$$

$$\begin{aligned} q^{[0]} &= \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \\ q^{[1]} &= \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \\ q^{[2]} &= \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ -2 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \end{bmatrix} + \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} \end{aligned}$$

  Note that these are the first three Taylor coefficients of the function $\begin{pmatrix} \sin(t) \\ \cos(t) \end{pmatrix}$ *at $t_0 = \pi$.*

A.3 Solving a System of Equations in AD

With this notation, we can illustrate how linear systems of equations in AD, i.e.,

$$B(t^{[0:K]}) \cdot x(t^{[0:K]}) \quad = \quad q(t^{[0:K]}),$$

can be solved.

Example 3 *For*

$$B(t) = \begin{pmatrix} \cos(t) & -\sin(t) \\ \sin(t) & \cos(t) \end{pmatrix}, \quad q(t) = \begin{pmatrix} \sin(t) \\ \cos(t) \end{pmatrix}$$

and $t^{[0:2]} := [\pi, \quad 1, \quad 0]$ *we obtain*

$$\Bigl[B^{[0]}, \quad B^{[1]}, \quad B^{[2]}\Bigr] \cdot \Bigl[x^{[0]}, \quad x^{[1]}, \quad x^{[2]}\Bigr] = \Bigl[q^{[0]}, \quad q^{[1]}, \quad q^{[2]}\Bigr]$$

i.e., using the rule of multiplication in AD we obtain the linear system

$$\begin{aligned} B^{[0]}x^{[0]} &= q^{[0]}, \\ B^{[1]}x^{[0]} + B^{[0]}x^{[1]} &= q^{[1]}, \\ B^{[2]}x^{[0]} + B^{[1]}x^{[1]} + B^{[0]}x^{[2]} &= q^{[2]}. \end{aligned}$$

If we insert the values

$$\begin{aligned} \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} x^{[0]} &= \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \\ \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} x^{[0]} + \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} x^{[1]} &= \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \\ \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} x^{[0]} + \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} x^{[1]} + \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} x^{[2]} &= \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} \end{aligned}$$

we can solve the system step by step, taking advantage of the block triangular form, and obtain the solution

$$\begin{bmatrix} x^{[0]}, & x^{[1]}, & x^{[2]} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -2 \end{bmatrix} \end{bmatrix},$$

which corresponds to the Taylor coefficients from

$$x(t) = \begin{pmatrix} \sin(2\,t) \\ \cos(2\,t) \end{pmatrix},$$

as considered in Example 2.
Analogously, nonlinear systems of equations and optimization problems can be solved by linearization, cf. Section 5.

References

References

[1] D. Estévez Schwarz, R. Lamour, InitDAEs documentation.
URL https://www.mathematik.hu-berlin.de/~lamour/software/python/InitDAE/html/

[2] E. Griepentrog, R. März, Differential-algebraic equations and their numerical treatment., Vol. 88 of Teubner-Texte zur Mathematik, B.G. Teubner Verlagsgesellschaft, Leipzig, 1986.

[3] D. Estévez Schwarz, R. Lamour, A new projector based decoupling of linear DAEs for monitoring singularities., Numer. Algorithms 73 (2) (2016) 535–565.

[4] D. Estévez Schwarz, R. Lamour, A new approach for computing consistent initial values and Taylor coefficients for DAEs using projector-based constrained optimization., Numer. Algorithms 78 (2) (2018) 355–377.

[5] K. Brenan, S. Campbell, L. Petzold, Numerical solution of initial-value problems in differential-algebraic equations. Unabridged, corr. republ., Classics in Applied Mathematics. 14. Philadelphia, PA: SIAM, Society for Industrial and Applied Mathematics, 1996.

[6] D. Estévez Schwarz, R. Lamour, Consistent initialization for higher-index DAEs using a projector based minimum-norm specification, Tech. Rep. 1, Institut für Mathematik, Humboldt-Universität zu Berlin (2016).

[7] S. F. Walter, L. Lehmann, Algorithmic differentiation in Python with AlgoPy, Journal of Computational Science 4 (5) (2013) 334 – 344. doi:http://dx.doi.org/10.1016/j.jocs.2011.10.007.
URL http://www.sciencedirect.com/science/article/pii/S1877750311001013

[8] Y. Chang, G. Corliss, ATOMFT: Solving ODEs and DAEs using Taylor series, Comput. Math. Appl. 28 (10-12) (1994) 209–233.

[9] S. L. Campbell, R. Hollenbeck, Automatic differentiation and implicit differential equations., Berz, Martin (ed.) et al., Computational differentiation: techniques, applications, and tools. Proceedings of the second international workshop on computational differentiation, February 12–14, 1996. Philadelphia, PA: SIAM. 215-227. (1996).

[10] J. D. Pryce, Solving high-index DAEs by Taylor series., Numer. Algorithms 19 (1-4) (1998) 195–211.

[11] J. D. Pryce, N. S. Nedialkov, G. Tan, X. Li, How AD can help solve differential-algebraic equations, Optimization Methods & Software 33 (4–6) (2018) 729–749.

[12] R. Barrio, Performance of the Taylor series method for ODEs/DAEs., Appl. Math. Comput. 163 (2) (2005) 525–545.

[13] R. Lamour, D. Monett Diaz, Index determination of DAEs – a wide field for automatic differentiation., Simos, Theodore E. (ed.) et al., ICNAAM 2009, Rethymno, Crete, Greece, September 18–22, 2009. Vol. 2. Melville, NY: American Institute of Physics (AIP). AIP Conference Proceedings 1168, 2, 727-730 (2009). (2009).

[14] D. Monett, R. Lamour, A. Griewank, Index determination in DAEs using the library indexnet and the ADOL-C package for algorithmic differentiation., in: Advances in automatic differentiation. Selected papers based on the presentations at the 5th international conference on automatic differentiation, Bonn, Germany, August 11–15, 2008, Berlin: Springer, 2008, pp. 247–257.

[15] D. Estévez Schwarz, R. Lamour, Projector based integration of DAEs with the Taylor series method using automatic differentiation, J. Comput. Appl. Math. 262 (2014) 62–72.

[16] D. Estévez Schwarz, R. Lamour, Monitoring singularities while integrating DAEs, in: S. Schöps, A. Bartel, M. Günther, E. ter Maten, P. Müller (Eds.), Progress in Differential-Algebraic Equations, Deskriptor 2013, Differential-Algebraic Equations Forum, Springer, 2014, pp. 73–95.

[17] R. Lamour, R. März, C. Tischendorf, Differential-algebraic equations: A projector based analysis, Differential-Algebraic Equations Forum 1. Berlin: Springer, 2013.

[18] Modelica 1.12.0.
URL `https://openmodelica.org`

[19] F. Mazzia, C. Magherini, Test set for initial value problems, release 2.4., Tech. rep., Department of Mathematics, University of Bari and INdAM, Research Unit of Bari (February 2008).
URL `http://pitagora.dm.uniba.it/~testset`

[20] S. L. Campbell, A general method for nonlinear descriptor systems: an example from robotic path control, Tech. rep., Department of Mathematics and Center for Research in Scientific Computing, North Carolina State University, CRSC Technical Report 090488-01 (October 1988).

[21] S. L. Campbell, E. Griepentrog, Solvability of general differential algebraic equations., SIAM J. Sci. Comput. 16 (2) (1995) 257–270. `doi:10.1137/0916017`.

[22] D. Estévez Schwarz, R. Lamour, R. März, The robotic arm problem causes a differential-algebraic equation featuring singularities, Tech. rep., Institut für Mathematik, Humboldt-Universität zu Berlin, in preparation (2019).

[23] R. Pulch, D. Estévez Schwarz, R. Lamour, Index-analysis for a method of lines discretising multirate partial differential algebraic equations., Appl. Numer. Math. 130 (2018) 51–69.

[24] A. Griewank, A. Walter, Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, 2nd Edition, SIAM, 2008.