Validated Simulation of Differential Algebraic Equations with Runge-Kutta Methods*†

Julien Alexandre dit Sandretto
julien.alexandre-dit-sandretto@ensta-paristech.fr

Alexandre Chapoutot
alexandre.chapoutot@ensta-paristech.fr

U2IS, ENSTA ParisTech, Université Paris-Saclay,
828 bd des Maréchaux, 91762 Palaiseau cedex France

Abstract

Differential Algebraic Equations (DAEs) are a general and implicit form of differential equations. DAEs are often used to represent physical systems such as dynamics of solids or chemical interactions. These equations are different from Ordinary Differential Equations (ODEs) in the sense that some of the dependent variables occur without their derivatives.

Validated simulation of ODEs has recently been the subject of different developments such as guaranteed Runge-Kutta integration schemes, both explicit and implicit. Not so different from solving an ODE, solving a DAE consists of searching for a consistent initial value and computing a trajectory. Nevertheless, DAEs are in generally much more difficult to solve than ODEs.

In this paper, we focus on the semi-explicit form of index one, called Hessenberg index-1 form. We propose a validated way to simulate this kind of differential equations. Finally, our method is applied to different examples in order to show its efficiency.

Keywords: Differential algebraic equations, Validated simulation.
AMS subject classifications: 65L80,65G20,65G40

1 Motivations

Our recent results on validated simulation of Ordinary Differential Equations (ODEs) with implicit Runge-Kutta schemes [2] leads consider more complex kinds of differential

*Submitted: November 20, 2015; Revised: January 26, 2016; Accepted: June 6, 2016.

†Partially funded by the Academic and Research Chair: "Complex Systems Engineering"- Ecole polytechnique ~ THALES ~ FX ~ DGA ~ DASSAULT AVIATION ~ DCNS Research ~ ENSTA ParisTech ~ Telecom ParisTech ~ Fondation ParisTech ~ FDO ENSTA

equations to solve. Indeed, we are able to simulate ODEs with interval parameters, which is one of the requirements for our solver of Differential Algebraic Equation (DAEs). We currently focus on DAEs in Hessenberg index-1 form, that is

$$\begin{cases} \dot{\mathbf{y}} = f(t, \mathbf{x}, \mathbf{y}) \\ 0 = g(t, \mathbf{x}, \mathbf{y}) \end{cases} \tag{1}$$

with $f : \mathbb{R} \times \mathbb{R}^m \times \mathbb{R}^n \mapsto \mathbb{R}^n$ and $g : \mathbb{R} \times \mathbb{R}^m \times \mathbb{R}^n \mapsto \mathbb{R}^m$.

In (1), $\mathbf{y} \in \mathbb{R}^n$ is the vector of differential variables, $\mathbf{x} \in \mathbb{R}^m$ is the vector of algebraic variables (without an expression for its derivative), and $\dot{\mathbf{y}}$ stands for the time derivative of $\mathbf{y}$. This kind of DAE is common and used by a majority of simulation tools such as Simulink and Modelica-like software. In this paper, we present a validated method to solve initial value problems given in the form of a DAE.

The article is organized as follows. In Section 2, a short description of our ODE solver, and by the same way, a state of the art of validated simulation of ODEs will be given. In Section 3, we review the literature on validated approaches for DAEs. In Section 4, we present our method in detail. In Section 5, we apply our solver on three different problems: a basic problem, a problem with a known solution and the classic pendulum problem. In Section 6, we discuss how to take into account additional constraints in IVPs for DAEs, before concluding in Section 7.

Notation

$\dot{y}$ denotes the time derivative of y, *i.e.*, $\frac{dy}{dt}$. a denotes a real value, while $\mathbf{a}$ represents a vector of real values. $[a]$ represents an interval, and $[\mathbf{a}]$ represents a vector of interval values. The midpoint of an interval $[x]$ is denoted by $\mathrm{m}([x])$. The function $\mathrm{Int}([\mathbf{x}])$ provides the interior of a box $[\mathbf{x}] = [\underline{\mathbf{x}}, \overline{\mathbf{x}}]$, such that $\underline{\mathbf{x}} \notin \mathrm{Int}([\mathbf{x}])$ and $\overline{\mathbf{x}} \notin \mathrm{Int}([\mathbf{x}])$. We denote the differential variables by y and the algebraic variables by x. The functions f and g are used to represent a differential algebraic equation system, with f the function given the time derivative of state variable and g the constraint on the algebraic variables. We also denote by $y(t, y_j)$ the exact solution of a differential equation at time t with a known value y_j at time t_j, in our case $t > t_j$. Sets will be represented by calligraphic letters such as $\mathcal{X}$ or $\mathcal{Y}$. The mathematical symbol $\exists!$ is used to denote the unique existential quantification (*i.e.*, "there is one and only one").

2 Validated Simulation of ODEs

A simulation of an ordinary differential equation consists of a discretization of time and an iterative and approximate computation of the state of the system, with the help of an integration scheme. An integration scheme $\Phi(t_{j+1}, t_j, \mathbf{y}_j)$, starting from an initial value $\mathbf{y}_j$ at time t_j and a step-size h produces an approximation $\mathbf{y}_{j+1}$ at time t_{j+1}, with $t_{j+1} - t_j = h$, of the solution $\mathbf{y}(t_{j+1}; \mathbf{y}_j)$. The associated local truncation error is defined by $\mathrm{LTE}_\Phi(t_{j+1}, t_j, \mathbf{y}_j) = \mathbf{y}(t_{j+1}; \mathbf{y}_j) - \Phi(t_{j+1}, t_j, \mathbf{y}_j)$. In this section, we briefly review how a validated simulation of an ODE is computed by bounding the LTE. We refer to [26] for a more detailed presentation.

2.1 Ordinary differential equations

An *initial value problem* (IVP) defined through an ODE is defined by

$$\dot{\mathbf{y}} = f(t, \mathbf{y}) \quad \text{with} \quad \mathbf{y}(0) = \mathbf{y}_0,\ \mathbf{y}_0 \in \mathbb{R}^n \quad \text{and} \quad t \in [0, t_{\text{end}}]\ . \tag{2}$$

In the classical approach [24, 26] to define a validated method for IVPs, each step of an integration scheme consists of two steps: *a priori enclosure* and *solution tightening*. Starting from a valid enclosure $[\mathbf{y}_j]$ at time t_j, the following two steps are applied.

Step 1. Compute an *a priori* enclosure $[\tilde{\mathbf{y}}_j]$ of the solution using Banach's theorem and the Picard-Lindelöf operator. This enclosure has the three major properties:

- $\mathbf{y}(t, [\mathbf{y}_j])$ is guaranteed to exist for all $t \in [t_j, t_{j+1}]$, *i.e.*, along the current step, and for all $\mathbf{y}_j \in [\mathbf{y}_j]$.
- $\mathbf{y}(t, [\mathbf{y}_j]) \subseteq [\tilde{\mathbf{y}}_j]$ for all $t \in [t_j, t_{j+1}]$.
- the step-size $h = t_{j+1} - t_j$ is as large as possible in terms of accuracy and existence proof for the IVP solution.

Step 2. Compute a tighter enclosure $[\mathbf{y}_{j+1}]$ such that $\mathbf{y}(t_{j+1}, [\mathbf{y}_j]) \subseteq [\mathbf{y}_{j+1}]$. The main issue in this phase is how to counteract the well known wrapping effect [24, 25, 26]. This phenomenon appears when we try to enclose a set with an interval vector (geometrically a box). The resulting overestimation creates a false dynamic for the next step, and, with accumulation, can lead to intervals with an unacceptably large width.

The different enclosures computed during one step are shown on Figure 1.

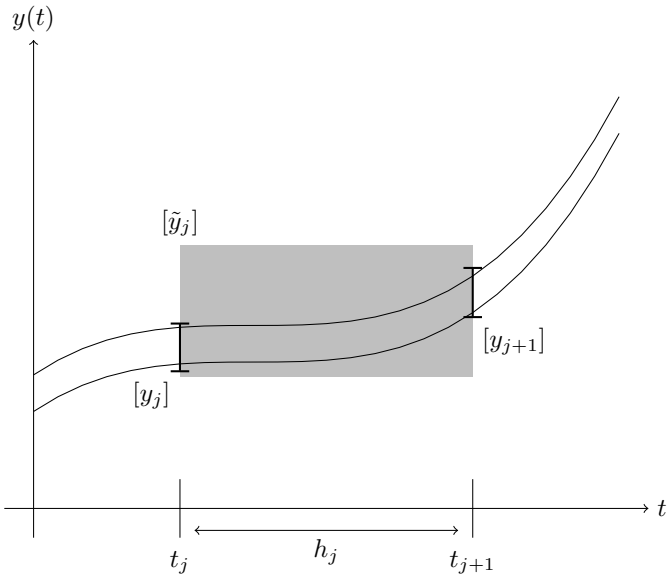


Figure 1: Enclosures appeared during one step

Some useful algorithms to perform these two steps are described in the following.

2.2 A priori solution enclosure

In order to compute an *a priori* enclosure, we use our interval version of Picard-Lindelöf operator. This operator is based on the following theorem.

Theorem 2.1 (Banach fixed-point theorem) *Let (K, d) be a complete metric space, given by a set K and a distance function $d : K \times K \mapsto \mathbb{R}$, and let $g : K \to K$ be a contraction; that is for all x, y in K there exists $c \in (0, 1)$ such that*

$$d\left(g(x), g(y)\right) \leq c \cdot d(x, y)$$

Then g has a unique fixed-point in K.

In the context of IVPs, we consider the space of continuously differentiable functions $C^1([t_j, t_{j+1}], \mathbb{R}^n)$ and the Picard-Lindelöf operator

$$P_f(y) = t \mapsto \mathbf{y}_j + \int_{t_n}^{t} f(s, \mathbf{y}(s))ds \ . \tag{3}$$

Note that this operator is associated with the integral form of Equation (2). So the solution of this operator is also the solution of Equation (2).

The Picard-Lindelöf operator is used to check the contraction of the solution on an integration step in order to prove the existence and the uniqueness of the solution of Equation (2) as stated by the Banach's fixed-point theorem. Furthermore, this operator is used to compute an enclosure of the solution of IVP over a time interval $[t_j, t_{j+1}]$.

2.2.1 Rectangular method for the a priori enclosure

Using interval analysis, and with a first order integration scheme, we can define a simple interval Picard-Lindelöf operator such that

$$P_f([\mathbf{r}]) = [\mathbf{y}_j] + [0, h] \cdot f([\mathbf{r}]), \tag{4}$$

with $h = t_{j+1} - t_j$ the step-size. Theorem 2.1 says that if we can find $[\mathbf{r}]$ such that $P_f([\mathbf{r}]) \subseteq [\mathbf{r}]$ then the operator is contracting, and Equation (2) has a unique solution. Furthermore,

$$\forall t \in [t_j, t_{j+1}], \quad \{\mathbf{y}(t; \mathbf{y}_j) : \forall \mathbf{y}_j \in [\mathbf{y}_j]\} \subseteq [\mathbf{r}],$$

then $[\mathbf{r}]$ is the *a priori* enclosure of the solution of Equation (2).

Note that the operator defined in Equation (4) can also define a contractor (in the sense of interval analysis [12]) on $[\mathbf{r}]$ after the contraction proved for P_f such that

$$[\mathbf{r}] \leftarrow [\mathbf{r}] \cap [\mathbf{y}_j] + [0, h].f([\mathbf{r}]) \ . \tag{5}$$

Hence, we can reduce the width of the *a priori* enclosure in order to increase the accuracy of the integration.

The operator defined in Equation (4) and its associated contractor defined in Equation (5) can be defined over a more accurate integration scheme (on condition that it is a guaranteed scheme like the interval rectangle rule). For example, the evaluation of $\int_{t_j}^{t} f(s)ds$ can be easily improved with a Taylor or a Runge-Kutta scheme (see [2]).

2.2.2 A priori enclosure with Taylor series

An interval version of the Taylor series ODE integration method is

$$[\mathbf{y}_{j+1}] \subset \sum_{k=0}^{N} f^{[k]}([\mathbf{y}_j])h^k + f^{[N+1]}([\tilde{\mathbf{y}}_j])h^{N+1}, \tag{6}$$

with $f^{[0]} = [\mathbf{y}_j]$, $f^{[1]} = f([\mathbf{y}_j]),\ldots,$ $f^{[k]} = \frac{1}{k}(\frac{\partial f^{[k-1]}}{\partial y} f)([\mathbf{y}_j])$.

By replacing h with the interval $[0, h]$, this scheme becomes an efficient Taylor Picard-Lindelöf operator, with a parametric order N such that

$$\mathbf{y}_{j+1}([t_j,t_{j+1}];[\mathbf{r}]) = \mathbf{y}_j + \sum_{k=0}^{N} f^{[k]}([\mathbf{y}_j])[0,h^k] + f^{[N+1]}([\mathbf{r}])[0,h^{N+1}] \ . \tag{7}$$

In consequence, if $[\mathbf{r}] \supseteq \mathbf{y}_{j+1}([t_j, t_{j+1}], [\mathbf{r}])$, then Equation (7) defines a contraction map and Theorem 2.1 can be applied.

In our tool, we use it at order 3 by default, which seems to be a good compromise between contractivity and computational efficiency.

Note that the scheme defined in Equation (6) is usually evaluated with a centered form for a more accurate result

$$[\mathbf{y}_{j+1}] \subset \sum_{k=0}^{N} f^{[k]}(\widehat{\mathbf{y}}_j)h^k + f^{[N+1]}([\tilde{\mathbf{y}}_j])h^{N+1} + \left(\sum_{k=0}^{N} J(f^{[k]},[\mathbf{y}_j])h^i)([\mathbf{y}_j] - \widehat{\mathbf{y}}_j)\right), \tag{8}$$

with $\widehat{\mathbf{y}}_j \in [\mathbf{y}_j]$. $J(f^{[k]}, [\mathbf{y}_j])$ is the Jacobian of $f^{[k]}$ evaluated at $[\mathbf{y}_j]$. This scheme can also be combined with a QR-factorization to increase stability and counteract the wrapping effect [26]. These two techniques, with a strong computational cost, can be replaced by using the affine arithmetic in the evaluation of $f^{[k]}$.

The Picard-Lindelöf operator, as defined in Equation (7), gives an *a priori* enclosure $[\mathbf{r}]$, using Theorem 2.1. If the Picard-Lindelöf operator is proven to be contracting on $[\mathbf{r}]$, we can then use this operator to contract the box $[\mathbf{r}]$ until a fixpoint is reached.

In our tool, the default contractor uses a Taylor expansion as follow

$$[\mathbf{r}] \cap \mathbf{y}_j + \sum_{k=0}^{N} f^{[k]}([\mathbf{y}_j])[0,h^k] + f^{[N+1]}([\mathbf{r}])[0,h^{N+1}].$$

It is important to contract this box as much as possible $[\mathbf{r}]$ because the Taylor remainder is function of $[\mathbf{r}]$, and the step-size is function of the Taylor remainder.

2.3 Tighter enclosure and truncation error

Suppose that Step 1 has been done for the current integration step, and that Step 1 has provided us a validated enclosure $[\tilde{\mathbf{y}}_j]$ such that

$$\mathbf{y}(t, t_j, [\mathbf{y}_j]) \subseteq [\tilde{\mathbf{y}}_j] \quad \forall t \in [t_j, t_{j+1}] \ .$$

In particular, we have $\mathbf{y}(t_{j+1}, t_j, [\mathbf{y}_j]) \subseteq [\tilde{\mathbf{y}}_j]$. The goal of Step 2 is thus to compute the tighter enclosure $[\mathbf{y}_{j+1}]$ such that

$$\mathbf{y}(t_{j+1}, t_j, [\mathbf{y}_j]) \subseteq [\mathbf{y}_{j+1}] \subseteq [\tilde{\mathbf{y}}_j] \ .$$

One way to do that consists of computing an approximate solution $\mathbf{y}_{j+1} \approx \mathbf{y}(t_{j+1}, t_j, [\mathbf{y}_j])$ with an integration scheme $\Phi(t_{j+1}, t_j, [\mathbf{y}_j])$, and then the associated local truncation error $\mathrm{LTE}_\Phi(t, t_j, [\mathbf{y}_j])$. Indeed, a guaranteed integration scheme has the property that there exists a time $\xi \in [t_j, t_{j+1}]$ such that

$$\mathbf{y}(t_{j+1}, t_j, [\mathbf{y}_j]) \subseteq \Phi(t_{j+1}, t_j, [\mathbf{y}_j]) + \mathrm{LTE}_\Phi(\xi, t_j, [\mathbf{y}_j]) \subseteq [\tilde{\mathbf{y}}_j] \ .$$

So $[\mathbf{y}_{j+1}] = \Phi(t_{j+1}, t_j, [\mathbf{y}_j]) + \mathrm{LTE}_\Phi(\xi, t_j, [\mathbf{y}_j])$ is an acceptable tight enclosure.

The guaranteed solution of IVP using interval arithmetic is mainly based on two kinds of methods to compute Φ:

1. Interval Taylor series methods [5, 13, 22, 23, 24, 25, 26, 36],
2. Interval Runge-Kutta methods [6, 7, 17].

The former is the oldest method used in this context, and until now, it is the most used method to solve Equation (2). The latter is more recent, see in particular [6, 7], but Runge-Kutta methods have many interesting properties, such as strong stability, that we would like to exploit in the context of validated solution of DAEs. The challenge lies in the computation of LTE_Φ.

2.4 Runge-Kutta methods

To obtain $\mathbf{y}_{n+1}$, a Runge-Kutta method computes s evaluations of f at predetermined time instants. The number s is the number of *stages* of a Runge-Kutta method. More precisely, a Runge-Kutta method is defined by

$$\mathbf{y}_{j+1} = \mathbf{y}_j + h \sum_{i=1}^{s} b_i \mathbf{k}_i \ , \tag{9}$$

with $\mathbf{k}_i$ defined by

$$\mathbf{k}_i = f\left(t_j + c_i h, \mathbf{y}_j + h \sum_{q=1}^{s} a_{iq} \mathbf{k}_q\right) \ . \tag{10}$$

The coefficients c_i, a_{iq} and b_i, for $i, q = 1, 2, \cdots, s$, fully characterize a Runge-Kutta method. They are usually synthesized in a *Butcher tableau* [11] of the form

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array}$$

A Runge-Kutta method can be

- *explicit*, *e.g.*, the classical Runge-Kutta method of order 4 given in Figure 2(a). In other words, the computation of an intermediate $\mathbf{k}_i$ only depends on the previous steps $\mathbf{k}_q$ for $q < i$.
- *diagonally implicit*, *e.g.*, a diagonally implicit method of order 4 given in Figure 2(b). In this case, the computation of an intermediate step $\mathbf{k}_i$ involves the value $\mathbf{k}_i$, in addition to $\mathbf{k}_q$ for $q < i$, and so non-linear systems in $\mathbf{k}_i$ must be solved.
- *fully implicit*, *e.g.*, the Runge-Kutta method with a Lobatto quadrature formula of order 4 given in Figure 2(c). In this last case, the computation of intermediate steps involves the solution of a non-linear system of equations in all the values $\mathbf{k}_i$ for $i = 1, 2, \cdots, s$. In this class of implicit methods, some of them have strong properties such as A-stability and stiffly accurate capability.

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

(a) RK4

$$\begin{array}{c|ccccc} \frac{1}{4} & \frac{1}{4} & & & & \\ \frac{3}{4} & \frac{1}{2} & \frac{1}{4} & & & \\ \frac{11}{20} & \frac{17}{50} & \frac{-1}{25} & \frac{1}{4} & & \\ \frac{1}{2} & \frac{371}{1360} & \frac{-137}{2720} & \frac{15}{544} & \frac{1}{4} & \\ 1 & \frac{25}{24} & \frac{-49}{48} & \frac{125}{16} & \frac{-85}{12} & \frac{1}{4} \\ \hline & \frac{25}{24} & \frac{-49}{48} & \frac{125}{16} & \frac{-85}{12} & \frac{1}{4} \end{array}$$

(b) SDIRK4

$$\begin{array}{c|ccc} 0 & \frac{1}{6} & -\frac{1}{3} & \frac{1}{6} \\ \frac{1}{2} & \frac{1}{6} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

(c) Lobatto3c

Figure 2: Different kinds of Runge-Kutta methods

Local truncation error

Recently, we defined a unified approach to express the LTE for explicit and implicit Runge-Kutta methods [2]. More precisely, following the notations of [20], for a Runge-Kutta of order p, we have

$$\begin{aligned} \mathrm{LTE}(t, \mathbf{y}(\xi)) &:= y(t_n; \mathbf{y}_{n-1}) - \mathbf{y}_n = \\ &\frac{h^{p+1}}{(p+1)!} \sum_{r(\tau)=p+1} \alpha(\tau) \left[1 - \gamma(\tau)\psi(\tau)\right] F(\tau)(\mathbf{y}(\xi)) \quad \xi \in [t_n, t_{n+1}] \end{aligned} \tag{11}$$

where

- τ is a *rooted tree*
- $F(\tau)$ is the *elementary differential* associated with τ
- $r(\tau)$ is the *order* of τ (number of nodes)
- $\gamma(\tau)$ is the *density*
- $\alpha(\tau)$ is the *number of equivalent trees*
- $\psi(\tau)$ a *function on trees*

Note that $\mathbf{y}(\xi)$ is a particular, and *a priori* unknown, solution of Equation (2) at a time instant ξ. This solution can be enclosed using Picard-Lindelöf operator presented in Section 2.2.

Wrapping effect

Runge-Kutta methods also suffer from the wrapping effect, as discussed before. The problem of reducing this latter has been studied in many different ways [28, 27]. One of the most known and effective is the *QR*-factorization [24]. This method improves the stability of the Taylor series in the Vnode-LP tool [26]. Another way is to modify the geometry of the enclosing set (parallelepipeds [16, 25], ellipsoids [28], convex polygons [31] and zonotopes [8, 34]).

In our work, an efficient affine arithmetic allows us to counteract the wrapping effect [2, 14].

3 Differential Algebraic Equations

We can distinguish at least two families of DAEs, fully implicit and semi-explicit.

3.1 Fully implicit DAEs

The first class of differential algebraic equations is the fully implicit ones. It is the most general representation of a differential system as:

$$f(t, \mathbf{y}, \dot{\mathbf{y}}, ...) = 0, \quad t_0 \leq t \leq t_{end} \tag{12}$$

3.2 Semi-explicit DAEs or in Hessenberg form

The second, and one of the most used DAE forms in science and engineering, is the semi-explicit DAEs, also called DAEs in Hessenberg form. In this formalism, the index is a differentiation index [10]. For example, the index 1 Hessenberg form is described by:

$$\begin{cases} \dot{\mathbf{y}} = f(t, \mathbf{x}, \mathbf{y}) \\ 0 = g(t, \mathbf{x}, \mathbf{y}) \end{cases} \tag{13}$$

where the Jacobian g_x is assumed to be non-singular for all t.

The index 2 Hessenberg form is described by:

$$\begin{cases} \dot{\mathbf{y}} = f(t, \mathbf{x}, \mathbf{y}) \\ 0 = g(t, \mathbf{y}) \end{cases} \tag{14}$$

where $g_y f_x$ is assumed to be non-singular for all t.

Since some of dependent variables occur without their derivatives, these differential systems are different from an ODE with an additive constraint.

3.3 Literature review

Although there exist many tools for solving DAEs in a numerical way (DAETS, Sundials, Mathematica) and some particular implementation embedded in simulation software (Simulink, Dymola), only few attempts have been done to obtain a guaranteed method.

We can notice an extension of Valencia-IVP [30], a tool for ODE validated simulation. The chosen approach starts with an approximation of the solution obtained by a numerical method (DASSL, DAETS) and try *a posteriori* to enclose the solution in a guaranteed way (with a Krawczyk iteration). The validity of this approach is not proven, in particular the fact that the existence test of the algebraic variable is separated from the state variable should not lead to a correct solution.

Another method [3] consists of computing the reachable set of DAEs by an error linearization and the help of zonotopes. This approach seems close to [30], and it is not clear on some points, such as existence and uniqueness proofs. Moreover, the results are not sufficiently explained to judge the quality of the method.

An alternative paper dealing with validated solution of DAEs available in the literature is based on Taylor models [21]. This method starts by computing a high-order polynomial approximating the solution of the ODE, and after that attempts to inflate it until an inclusion test validates. The approach presented is then different

from our method. Unfortunately, none of these two latter approaches seems to be continued.

Finally, and more recently, one important work appears in [32, 33]. In these papers, the authors present the analysis and the computation of bounds by using i) an existence and uniqueness test for the solutions of DAEs, based on Hansen-Sengupta, and ii) sufficient conditions for two functions to enclose lower and upper bounds of the solution. Another method is presented unifying these two steps. Regrettably, the algorithms are not sufficiently clearly described to allow a thorough analysis and comparison. The results presented are also difficult to assess. Nevertheless, we are able to make few remarks. First, the presented approach is not completely validated (as honestly said in the paper). Second, if it is well known that the Hansen-Sengupta test is more efficient than the Krawczyk one [18], in [32] it is used in the non-parametric version, while we advocate to use Krawczyk in the parametric version. Indeed, Hansen-Sengupta requires solving a linear interval system, which can be a time consuming function of problem size and done many times at each integration step (until conditions of Theorem 2.1 are obtained). Moreover, the operator is applied to a non linear function with state variables as interval parameters, which can be large due to initial state, and the use of a parametric version is then necessary to obtain a sufficiently sharp solution (see Section 4.1).

It is important to remark that the problem of existence and uniqueness of the solution is a common issue considered in the literature, whether in the validated approaches, presented above, or in the numerical field, for which consistent initialization is one of the main issues [35].

4 Our Method for Validated Simulation of DAE

In this work, we present a method to solve the *initial value problems* written in index-1 Hessenberg form:

$$\begin{cases} \dot{\mathbf{y}} = f(t, \mathbf{x}, \mathbf{y}) \\ 0 = g(t, \mathbf{x}, \mathbf{y}) \end{cases} \quad \text{with} \quad \mathbf{y}(0) \in [\mathbf{y}_0] \quad \text{and} \quad \mathbf{x}(0) \in [\mathbf{x}_0] \ . \tag{15}$$

In Section 2.1, we reviewed the classical two step method presented originally by Lohner [24], and used by the community of ODE validated integration. We used the same approach for our DAE integration method, as [33]. The two steps are:

- Compute an *a priori* enclosure of all the solutions of the DAE on an integration step $[t_j, t_j + h]$ with a novel Picard-like operator;
- Refine this enclosure at $t_j + h$ with the Runge-Kutta integration scheme and contractors.

In the next subsections, we will present these two steps in detail.

4.1 New Picard-like operator

For an ordinary differential equation with interval parameters (similar to differential inclusion) described by

$$\dot{\mathbf{y}} = f(t, \mathbf{y}, \mathbf{p}) \quad \text{with} \quad \mathbf{y}(0) \in [\mathbf{y}_0] \quad \text{and} \quad \mathbf{p} \in [\mathbf{p}] \ , \tag{16}$$

we use the Picard-Lindelöf operator. This operator, based on the Theorem 2.1 and defined in Equation (3), allows one to compute the *a priori* enclosure $[\tilde{\mathbf{y}}_j]$ such that

$$\forall t \in [t_j, t_{j+1}], \quad \{\mathbf{y}(t; \mathbf{y}_j) : \forall \mathbf{y}_j \in [\mathbf{y}_j],\ \forall \mathbf{p} \in [\mathbf{p}]\} \subseteq [\tilde{\mathbf{y}}_j] \ .$$

In the case of a DAE expressed by Equation (15), the further issue is to compute the *a priori* enclosure $[\tilde{\mathbf{x}}_j]$ such that

$$\forall t \in [t_j, t_{j+1}], \quad \{\mathbf{x}(t; \mathbf{y}_j) : \forall \mathbf{y}_j \in [\mathbf{y}_j]\} \subseteq [\tilde{\mathbf{x}}_j],$$

under the constraint $g(\mathbf{x}(t), \mathbf{y}(t)) = 0, \quad \forall t \in [t_j, t_{j+1}]$.

4.1.1 Analysis of existence and uniqueness

If we assume that $\frac{\partial g}{\partial x}$ is locally non-singular, we are theoretically able to find the unique $\mathbf{x} = \psi(\mathbf{y})$ (with the help of the implicit function theorem), and then we can write $\dot{\mathbf{y}} = f(t, \psi(\mathbf{y}), \mathbf{y})$. Finally, we could apply Picard-Lindelöf to f in order to prove *existence and uniqueness* of the solution. Of course, it is not feasible nor realistic in general, because ψ is unknown. Anyway, this approach leads us to the following solution.

We propose a theorem based on the Frobenius theorem with a set membership view. The Frobenius theorem is too long to be given completely here, but it is available in [15].

Theorem 4.1 (Part of the Frobenius theorem) *Let $\mathcal{X}$ and $\mathcal{Y}$ be Banach spaces, and let $\mathcal{A} \subset \mathcal{X}$, and $\mathcal{B} \subset \mathcal{Y}$ be open sets. Let $F : \mathcal{A} \times \mathcal{B} \to \mathcal{L}(\mathcal{X}, \mathcal{Y})$ be a continuously differentiable function of the Cartesian product (which inherits a differentiable structure from its inclusion into $\mathcal{X} \times \mathcal{Y}$) into the space $\mathcal{L}(\mathcal{X}, \mathcal{Y})$ of continuous linear transformations of $\mathcal{X}$ into $\mathcal{Y}$. A differentiable mapping $u : \mathcal{A} \to \mathcal{B}$ is a solution of the differential equation*

$$\dot{y} = F(x, y) \tag{17}$$

if $\dot{u}(x) = F(x, u(x))$ for all $x \in \mathcal{A}$.

Equation (17) is completely integrable if for each $(x_0, y_0) \in \mathcal{A} \times \mathcal{B}$, there is a neighborhood $\mathcal{U}$ of x_0 such that Equation (17) has a unique solution $u(x)$ defined on $\mathcal{U}$ such that $u(x_0) = y_0$.

Theorem 4.2 (Banach space version of Implicit Function Theorem) *Let $\mathcal{X}$, $\mathcal{Y}, \mathcal{Z}$ be Banach spaces. Let the mapping $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ be continuously Fréchet differentiable. If $(x_0, y_0) \in \mathcal{X} \times \mathcal{Y}, f(x_0, y_0) = 0$, and $y \mapsto Df(x_0, y_0)(0, y)$ is a Banach space isomorphism from $\mathcal{Y}$ onto $\mathcal{Z}$, then there exist neighborhoods $\mathcal{U}$ of x_0 and $\mathcal{V}$ of y_0 and a Fréchet differentiable function $g : \mathcal{U} \to \mathcal{V}$ such that $f(x, g(x)) = 0$ and $f(x, y) = 0$ if and only if $y = g(x)$, for all $(x, y) \in \mathcal{U} \times \mathcal{V}$.*

Proposition 4.1 (Set membership view of Frobenius) *Let $\mathcal{A}$ and $\mathcal{B}$ be two Banach spaces and let g, f be two continuously differentiable function from the Cartesian product $\mathcal{A} \times \mathcal{B}$ into the set $\mathcal{L}(\mathcal{A}, \mathcal{B})$ of continuous linear transformations of $\mathcal{A}$ into $\mathcal{B}$, defined by:*

$$\dot{y} = f(t, x, y) \tag{18}$$

and

$$g(x, y) = 0. \tag{19}$$

Let $\mathcal{X} \subset \mathcal{A}$ and $\mathcal{Y} \subset \mathcal{B}$. If $\forall y \in \mathcal{Y}, \exists!\ x \in \mathcal{X}$ satisfying Equation (19) and $\forall x \in \mathcal{X}, \exists!\ y \in \mathcal{Y}$ satisfying Equation (18), simultaneously, then Equation (18) is completely integrable and the solution exists and is unique in $\mathcal{Y}$.

Proof: If $\forall y \in \mathcal{Y}, \exists!\, x \in \mathcal{X} : g(x,y) = 0$, then the Implicit Function Theorem proves that there is a unique function $x = \psi(y)$ for all $(x,y) \in \mathcal{A} \times \mathcal{B}$. Then, the differential Equation (18) becomes $\dot{y} = f(t, \psi(y), y)$ where $\psi(y)$ maps $\mathcal{Y}$ into $\mathcal{X}$. Finally, the statement $\forall x \in \mathcal{X}, \exists!\, y \in \mathcal{Y} : \dot{y} = f(t,y,x)$ implies, with the help of Frobenius theorem, that there exists a unique mapping solution of Equation (18) in $\mathcal{Y}$. □

4.1.2 Adapted to DAEs

A direct translation of Proposition 4.1 to our problem gives:

Proposition 4.2 *Let $[\tilde{\mathbf{y}}]$ be an enclosure, validated by the Picard-Lindelöf operator, of the solution of $\dot{\mathbf{y}} \in f(t, [\tilde{\mathbf{x}}], \mathbf{y})$ and let $[\tilde{\mathbf{x}}]$ be defined such that for each $\mathbf{y} \in [\tilde{\mathbf{y}}], \exists!\, \mathbf{x} \in [\tilde{\mathbf{x}}] : g(\mathbf{x}, \mathbf{y}) = 0$. Then $\exists!\, \psi$ on the neighborhood of $[\tilde{\mathbf{x}}]$, and the solution of the DAE exists and is unique in $[\tilde{\mathbf{y}}]$. In addition, the algebraic variable is enclosed by $[\tilde{\mathbf{x}}]$.*

Proof: Direct application of Proposition 4.1. □

4.1.3 The operator

Finally, with Proposition 4.2, we define a novel operator, which we call Picard-Krawczyk $\mathcal{PK}$:

$$\text{If} \quad \begin{pmatrix} \mathcal{P}([\tilde{\mathbf{y}}], [\tilde{\mathbf{x}}]) \\ \mathcal{K}([\tilde{\mathbf{y}}], [\tilde{\mathbf{x}}]) \end{pmatrix} \subset \text{Int} \begin{pmatrix} [\tilde{\mathbf{y}}] \\ [\tilde{\mathbf{x}}] \end{pmatrix} \quad \text{then} \quad \exists! \quad \text{solution of DAE in } [\tilde{\mathbf{y}}] \tag{20}$$

with

- $\mathcal{P}$ a Picard-Lindelöf operator for the differential inclusion $\dot{\mathbf{y}} \in f([\tilde{\mathbf{x}}], \mathbf{y})$
- $\mathcal{K}$ a parametric preconditioned Krawczyk operator for the constraint $g(\mathbf{x}, \mathbf{y}) = 0, \forall \mathbf{y} \in [\tilde{\mathbf{y}}]$

In more detail, these operators are given by:

Picard-Lindelöf operator with Taylor ($N = 3$):

$$\mathcal{P}_f([\mathbf{y}_j], [\mathbf{x}_j], [\mathbf{r}], [\tilde{\mathbf{x}}], h) = [\mathbf{y}_j] + \sum_{k=0}^{N} f^{[k]}([\mathbf{x}_j], [\mathbf{y}_j])[0, h^k] + f^{[N+1]}([\tilde{\mathbf{x}}], [\mathbf{r}])[0, h^{N+1}] \ . \tag{21}$$

If $\mathcal{P}_f([\mathbf{y}_j], [\mathbf{x}_j], [\mathbf{r}], [\tilde{\mathbf{x}}], h) \subset \text{Int}([\mathbf{r}])$ then f is integrable and $[\mathbf{y}_{j+1}] \subset [\mathbf{r}]$.

Parametric preconditioned Krawczyk operator:

$$\begin{aligned} \mathcal{K}_g([\tilde{\mathbf{y}}], [\mathbf{r}]) = \text{m}([\mathbf{r}]) - C g(\text{m}([\mathbf{r}]), \text{m}([\tilde{\mathbf{y}}])) - \\ (C \frac{\partial g}{\partial x}([\mathbf{r}], [\tilde{\mathbf{y}}]) - I)([\mathbf{r}] - \text{m}([\mathbf{r}])) - \\ C \frac{\partial g}{\partial y}(\text{m}([\mathbf{r}]), [\tilde{\mathbf{y}}])([\tilde{\mathbf{y}}] - \text{m}([\tilde{\mathbf{y}}])) \end{aligned} \tag{22}$$

where C is a preconditioning matrix. If $\mathcal{K}_g([\tilde{\mathbf{y}}], [\mathbf{r}]) \subset \text{Int}([\mathbf{r}])$ then for each $\mathbf{y} \in [\tilde{\mathbf{y}}]$ there exists one and only one $\mathbf{x} \in [\mathbf{r}] : g(\mathbf{x}, \mathbf{y}) = 0$.

Parametric preconditioned Krawczyk operator in hybrid form:
A hybrid form [19] is also available:

$$\begin{aligned}
[\mathbf{s}] &= Cg(\mathrm{m}([\mathbf{r}]), \mathrm{m}([\tilde{\mathbf{y}}])) + C\frac{\partial g}{\partial y}(\mathrm{m}([\mathbf{r}]), [\tilde{\mathbf{y}}])([\tilde{\mathbf{y}}] - \mathrm{m}([\tilde{\mathbf{y}}])) \\
[\mathbf{s}] &= [\mathbf{s}] \cap (Cg(\mathrm{m}([\mathbf{r}]), [\tilde{\mathbf{y}}])) \\
\mathcal{K}_g([\tilde{\mathbf{y}}], [\mathbf{r}]) &= \mathrm{m}([\mathbf{r}]) - [\mathbf{s}] - (C\frac{\partial g}{\partial x}([\mathbf{r}], [\tilde{\mathbf{y}}]) - I)([\mathbf{r}] - \mathrm{m}([\mathbf{r}]))
\end{aligned}$$

If $\mathcal{K}_g([\tilde{\mathbf{y}}], [\mathbf{r}]) \subset \mathrm{Int}([\mathbf{r}])$ then for each $\mathbf{y} \in [\tilde{\mathbf{y}}]$ there exists one and only one $\mathbf{x} \in [\mathbf{r}]$: $g(\mathbf{x}, \mathbf{y}) = 0$.

4.1.4 Algorithm implementing the Picard-Krawczyk operator

We propose an algorithm implementing this Picard-Krawczyk operator to compute the enclosures of the algebraic and state variables simultaneously in Algorithm 1. The inputs are the initial estimation for enclosures $[\tilde{x}]^0$, $[\tilde{y}]^0$, a tolerance on LTE Tol and a guessed step size h. The outputs are either a success flag with the validated enclosures $[\tilde{x}]^1$, $[\tilde{y}]^1$ and the already computed LTE, or a fail flag. We iterate two times the dimension of the problem $(m + n)$ to prapagate the inflation.

Algorithm 1 Compute the *a priori* enclosures

Require: $[\tilde{\mathbf{x}}]^0$, $[\tilde{\mathbf{y}}]^0$, Tol, h, iter $= 0$
  $[\tilde{\mathbf{x}}]^1 = \mathcal{K}([\tilde{\mathbf{y}}]^0, [\tilde{\mathbf{x}}]^0)$ // 4.1.3
  $[\tilde{\mathbf{y}}]^1 = \mathcal{P}([\tilde{\mathbf{y}}]^0, [\tilde{\mathbf{x}}]^0)$ // 4.1.3
  while $([\tilde{\mathbf{x}}]^1 \not\subset [\tilde{\mathbf{x}}]^0)$ **and** $([\tilde{\mathbf{y}}]^1 \not\subset [\tilde{\mathbf{y}}]^0)$ **and** (iter $< 2(m + n)$) **do**
    iter $=$ iter $+ 1$
    $[\tilde{\mathbf{y}}]^0 = [\tilde{\mathbf{y}}]^1 \pm 1\%$
    $[\tilde{\mathbf{x}}]^0 = [\tilde{\mathbf{x}}]^1 \pm 1\%$
    $[\tilde{\mathbf{x}}]^1 = \mathcal{K}([\tilde{\mathbf{y}}]^0, [\tilde{\mathbf{x}}]^0)$ // 4.1.3
    $[\tilde{\mathbf{y}}]^1 = \mathcal{P}([\tilde{\mathbf{y}}]^0, [\tilde{\mathbf{x}}]^0)$ // 4.1.3
  end while
  if $([\tilde{\mathbf{x}}]^1 \subset [\tilde{\mathbf{x}}]^0)$ and $([\tilde{\mathbf{y}}]^1 \subset [\tilde{\mathbf{y}}]^0)$ **then** // conditions of 4.2 obtained
    LTE $=$ LTE$([\tilde{\mathbf{x}}]^1, [\tilde{\mathbf{y}}]^1)$ // 2.4
    if LTE $<$ Tol **then** // Acceptable LTE
      return SUCCESS($[\tilde{\mathbf{x}}]^1$, $[\tilde{\mathbf{y}}]^1$, LTE)
    end if
  end if
  return FAILED // No enclosures found with the inputs

4.2 Contractors

After the guaranteed enclosures on $[t, t + h]$ are obtained, we can contract these enclosures around $t + h$, because obviously $\mathbf{x}(t + h) \subset [\tilde{\mathbf{x}}]$ and $\mathbf{y}(t + h) \subset [\tilde{\mathbf{y}}]$. First, it is more efficient to start by the contraction of the state variable $\mathbf{y}$, which can be strongly refined by the well chosen integration scheme. The differential inclusion to integrate is given by: $\dot{\mathbf{y}} \in f(t, [\tilde{\mathbf{x}}], \mathbf{y})$. This latter is often stiff (this assumption will be verified in

section 5.1) and has interval coefficients. As demonstrated in [2], an implicit Runge-Kutta scheme is efficient for this kind of interval parametrized differential equation. In the family of Implicit Runge-Kutta (IRK), Radau IIA is one of the more powerful methods of order 3 for stiff problems. Its Butcher tableau is given in Figure 3.

This IRK method attains an order three with two stages, it is fully implicit as shown in its tableau (Figure 3) and A-stable. The corresponding local truncation error (LTE) can be computed using the Butcher trees as shown in [2].

1/3	5/12	-1/12
1	3/4	1/4
	3/4	1/4

Figure 3: Butcher Tableau of Radau IIA order 3 (an Implicit Runge-Kutta method)

Second, with a tightened state variable, we are able to refine the algebraic variable $\mathbf{x}$ under the constraint $g(t, \mathbf{x}, \mathbf{y}) = 0, \forall \mathbf{y} \in [\tilde{\mathbf{y}}]$. For this task, we combine the Krawczyk operator from Section 4.1 and a forward/backward contractor coming from constraint programming. The Forward/Backward contractor (also called HC4-Revise [4], denoted by $\text{FwdBwd}_{\text{constraint}}$ in the following algorithms) can contract a box w.r.t. a single constraint such that no solution of the constraint is lost in the box. Using a tree representation of the constraint for accelerating the contraction, this contractor isolates every occurrence $\mathbf{x}_i$ in the expression and performs a natural evaluation of the corresponding function to contract $[\mathbf{x}_i]$. This combination can be easily done by the contractor programming view of the tool DynIbex[1].

Finally, these two obtained contractors are embedded in a fixpoint presented in Algorithm 2. The inputs of this algorithm are the enclosures and the LTE computed with Algorithm 1. The outputs are the guaranteed enclosures of the state and algebraic solutions at the end of current time step.

Algorithm 2 Contract the *a priori* enclosures around solution

Require: $[\tilde{\mathbf{x}}]$, $[\tilde{\mathbf{y}}]$, LTE
  $[\mathbf{x}_{j+1}] = [\tilde{\mathbf{x}}]$, $[\mathbf{y}_{j+1}] = [\tilde{\mathbf{y}}]$
  while ($[\mathbf{x}_{j+1}]$ **or** $[\mathbf{y}_{j+1}]$ is improved) **do** // fix point
    $[\mathbf{y}_{j+1}] \cap = (\text{RADAU3}_f([\mathbf{x}_{j+1}], [\mathbf{y}_{j+1}]) + \text{LTE})$ // 2.4 and 3
    $[\mathbf{x}_{j+1}] \cap = (\text{FwdBwd}_g([\mathbf{x}_{j+1}], [\mathbf{y}_{j+1}]) \cap \mathcal{K}_g([\mathbf{x}_{j+1}], [\mathbf{y}_{j+1}]))$ // [4] and 4.1.3
  end while

4.3 Complete algorithm

The algorithm for a validated simulation of a DAE in index-1 Hessenberg form is given in a simplified way, without the step-size controller (available in [20]), in Algorithm 3. The inputs are initial states, a time to reach T_{final}, a given minimal step-size $h_{\min}$ and a tolerance Tol. The algorithm builds a list of data computed at each integration step, as described in Section 2.1.

[1] `http://perso.ensta-paristech.fr/~chapoutot/dynibex/`

Algorithm 3 Simulation of DAE

Require: $[y(0)]$, $[x(0)]$, T_{final}, $h_{\min}$, Tol // Problem statement
  $t = 0$
  while $(t < T_{\text{final}})$ **do** // Until desired time
    $[\tilde{x}] = x(t), [\tilde{y}] = y(t)$
    Computation of $[\tilde{x}], [\tilde{y}]$, LTE // Algorithm 1
    if SUCCESS **then**
      Computation of $[x(t+h)], [y(t+h)]$ // Algorithm 2
      Store $[t, t+h], [\tilde{x}], [\tilde{y}], [x(t+h)], [y(t+h)]$ in a list
      $t = t + h$ // Next integration step
    else if $(h > h_{\min})$ **then**
      $h = h/2$ // Reducing of step-size
    else
      Return FAILED // Integration failed (change initial states, Tol or $h_{\min}$)
    end if
  end while

5 Examples

We apply our algorithm to three examples. The first one is quite basic, but allows us to show the issue appearing in DAE simulation and the results obtained by our tool. This example is sufficient to point out the problem of stiffness generated by DAE problems. The second example is interesting because it has a known exact solution and allows us to verify the exactness of our method and judge the results with respect to the best existing numerical method. Finally, the third example is the classical pendulum problem, which permits us to highlight the efficiency of our method.

5.1 Basic example

We start our experimentations with a basic example in one dimension for the state variable and one dimension for the algebraic variable:

$$\begin{cases} 0 = (y+1)x + 2 \\ y' = y + x + 1 \end{cases} \qquad y(0) = 1.0 \quad \text{and} \quad x(0) \in [-2.0, 2.0]$$

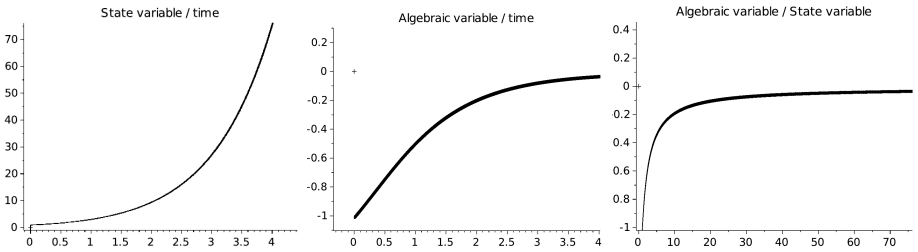
We perform a simulation until $T_{\text{final}} = 4$ seconds with a desired tolerance on the local error Tol=10^{-16}. The initial value consistency is checked with Krawczyk which leads to $x(0) = -1$. The computation takes between 16 and 30 seconds depending on the computer. Our tool provides three files containing:

- The values of state variable enclosure w.r.t. time under the form: $[y(t)]; t$
- The values of algebraic variable enclosure w.r.t. time under the form: $[x(t)]; t$
- The log reported in Table 1

With the files containing state and algebraic values w.r.t. time, we are able to plot three figures, given in Figure 4. In Figure 4(a) and Figure 4(b), it is apparent that even if the state variable and algebraic variable evolve exponentially but quite slowly, algebraic variable evolves in a stiff way w.r.t. state variable, Figure 4(c). In general, DAEs lead to a stiff problem, which is the motivation for using the RADAU IIA method. This method is known for its efficiency on stiff problems.

Log.txt file	Description
`Solution at` $t = 4.00000$:	Final time reached
$([76.2255, 76.2299])$	Solution for state variable
`Diameter:` (0.00447355)	Diameter of the solution enclosure
`Rejected Picard:` 2	Number of step rejected by Algorithm 1
`Accepted Picard:` 19123	Number of step accepted by Algorithm 1
`Step min:` 0.000196651	Time step minimum (h)
`Step max:` 0.00025	Time step maximum (h)
`Truncature error max:` $2.1434 \cdot 10^{-15}$	Maximum LTE during simulation

Table 1: Log file and its description



(a) (b) (c)

Figure 4: Plot of state variable w.r.t. time (a), algebraic variable w.r.t. time (b) and algebraic variable w.r.t. state variable (c) (for Problem 5.1)

5.2 Example with exact solution

The second example is a little more complex. It is described by the differential equation:

$$y' = \begin{cases} -y_2 y_1 - (1+y_2)x_0 \\ y_2 y_0 - (1+y_2)x_1 \\ 1 \end{cases}$$

and the algebraic relation:

$$\begin{cases} (y_0 - x_1)/5 - \cos(y_2^2)/2 = 0 \\ (y_1 + x_0)/5 - \sin(y_2^2)/2 = 0 \end{cases}$$

$$\text{with} \quad y(0) = \begin{pmatrix} 5 \\ 1 \\ 0 \end{pmatrix} \quad \text{and} \quad x(0) \in \begin{pmatrix} [-1,-1] \\ [-10^{-14}, 10^{-14}] \end{pmatrix}$$

This problem is interesting because it has a known exact solution:

$$\begin{cases} y_0 = \sin(t) + 5\cos((t^2)/2.0) \\ y_1 = \cos(t) + 5\sin((t^2)/2.0) \\ x_0 = -\cos(t) \\ x_1 = \sin(t) \end{cases}$$

We perform a simulation until $T_{\text{final}} = 2$ seconds with a desired tolerance $\text{Tol} = 10^{-22}$. This tolerance is chosen very small (below machine precision) in order to force

the LTE to be as small as possible. The initial value consistency is checked with Krawczyk which leads to

$$x(0) \in \begin{pmatrix} [-1,-1] \\ [-10^{-18}, 10^{-18}] \end{pmatrix}.$$

The results at final time are given in Table 2. These results allow us to verify the enclosure of the exact solution by the solution provided by our tool. In fact, we perform a verification of inclusion on the fly, that is to say, at each step of the simulation. Moreover, at $t = 2$ seconds, the diameter of our solution is smaller than 0.00056 when one of the best numerical methods (without guarantee) [1], the Extended Block Backward Differentiation Formula (EBBDF) of order 4, provides a result at 0.0002 from the exact value, which is comparable to our results (see Table 2).

Our method	Exact	EBBDF
$[-1.17172, -1.17116]$	-1.171439444	-1.171279
$[4.13013, 4.13054]$	4.130338864	4.130540
$[0.415948, 0.416352]$	0.4161470936	0.416035
$[0.909204, 0.909388]$	0.9092973092	0.909322

Table 2: Solution at $t = 2$ with our method, exact value computation and EBBDF results (from [1]) for Problem 5.2

5.3 Classical problem: Pendulum

Finally, the last example is the well known pendulum, expressed in index 1 Hessenberg form. The problem can be described by Figure 5.

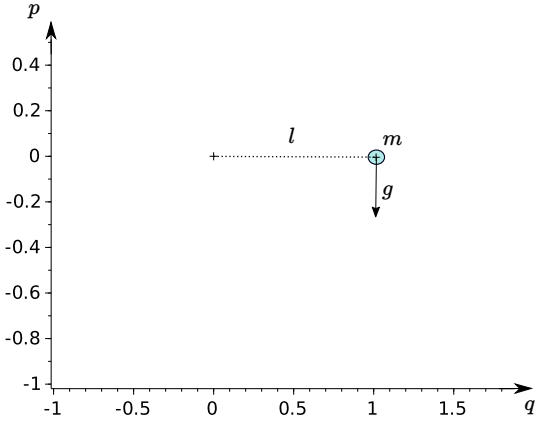


Figure 5: Problem view of Example 5.3

A ball of mass m, suspended to a massless rod of length ℓ, under gravity g is

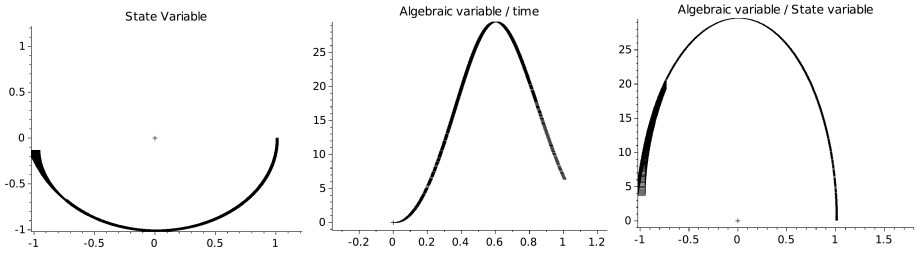
defined by coordinates (p, q) such that:

$$\begin{cases} p' = u \\ q' = v \\ mu' = -p\lambda \\ mv' = -q\lambda - g \end{cases}$$

with u and v the velocity of pendulum, and λ the force in the rod, and the algebraic relation:

$$m(u^2 + v^2) - gq - \ell^2\lambda = 0 \quad \text{with} \quad (p, q, u, v)_0 = (1, 0, 0, 0) \quad \text{and} \quad \lambda_0 \in [-0.1, 0.1]$$

We perform a simulation until $T_{\text{final}} = 1$ second with a desired tolerance $\text{Tol} = 10^{-10}$. The initial value consistency is checked with Krawczyk, which leads to $\lambda(0) = 0$. The computation takes less than 20 seconds on a dual core computer. With the files containing state and algebraic values w.r.t. time, we are able to plot three figures, given in Figure 6.



(a) (b) (c)

Figure 6: Plot of state variable p w.r.t. q (a), algebraic variable w.r.t. time (b) and algebraic variable w.r.t. state variable p (c) (for Problem 5.3)

5.4 Discussion

We see through these three examples that our method is correct (Example 5.2), efficient in comparison to the existing non guaranteed methods (Example 5.2), able to solve a classical problem (Example 5.3), while providing statistics on the computation progress (Example 5.1). Nevertheless, we can also conclude that the diameter of solution grows quickly and that time spent for the computation is sometimes significant. In the following section, we present an add-on in order to improve our results.

6 Additive Contractors

Our tool is based on contractor programming [12]. This means that we are able to add any contractors to Algorithm 2 to take into account constraints other than $g(t, x, y) = 0$. In the case of physical systems, constraints can appear from the context such as energy conservation, mechanical constraints, etc. This approach is similar to [9]. In the latter, the authors use the energy conservation as constraint in order to improve the precision of a numerical simulation. More interestingly, some constraints

come directly from the Pantelides algorithm [29] which is used to reduce the index of DAEs.

For example, in the case of Pendulum (Example 5.3), the Pantelides algorithm is used to obtain $m(u^2+v^2) - gq - \ell^2\lambda = 0$ by differentiation of the circle equation $p^2+q^2-\ell^2=0$:

$$\begin{cases} p^2+q^2-\ell^2 = 0 \\ pu+qv = 0 \\ m(u^2+v^2) - gq^2 - \ell^2 p = 0 \end{cases}$$

These three additive constraints can be used to generate a forward/backward contractor (with a propagation principle between constraints) for both state and algebraic variables. This contractor improves the result of our algorithm by changing Algorithm 2 to Algorithm 4.

Algorithm 4 Contract the *a priori* enclosures around solution

Require: $[\tilde{x}]$, $[\tilde{y}]$, LTE
  $[x_{j+1}] = [\tilde{x}]$, $[y_{j+1}] = [\tilde{y}]$
  while ($[x_{j+1}]$ **or** $[y_{j+1}]$ is improved) **do**
    $[y_{j+1}]\cap = (\text{RADAU3}_f([x_{j+1}],[y_{j+1}]) + \text{LTE})$
    $[x_{j+1}]\cap = (\text{FwdBwd}_g([x_{j+1}],[y_{j+1}]) \cap \mathcal{K}_g([x_{j+1}],[y_{j+1}]))$
    $([x_{j+1}];[y_{j+1}])\cap = \text{FwdBwd}_{ctc}(([x_{j+1}];[y_{j+1}]))$
  end while

Our tool is applied with Algorithm 2 and Algorithm 4 to the Pendulum problem for a simulation until $T_{\text{final}} = 1.6$ seconds and for a desired tolerance $\text{Tol} = 10^{-18}$. The results are shown in Figure 7 and Figure 8. It is obvious that the addition of contractors has improved the simulation by reducing the size (and thus the pessimism) of the enclosures, approximately 50%, and even the time computation.

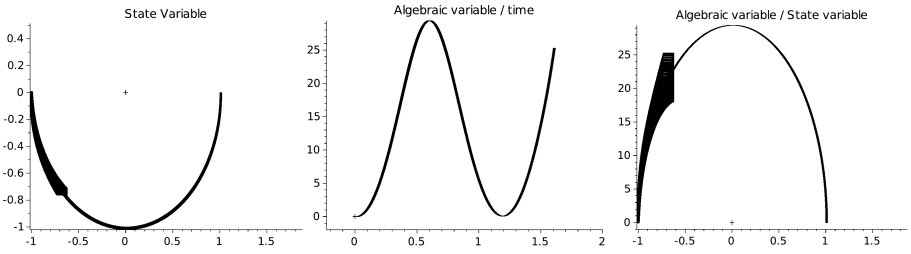


Figure 7: Plot of state variable p w.r.t. q (a), algebraic variable w.r.t. time (b) and algebraic variable w.r.t. state variable p (c) (for Problem 5.3) - with Algorithm 2: computation time 28 minutes

7 Conclusions and Future Work

To conclude, we present in this paper our method for the validated simulation of differential algebraic equations. The main issue being the existence and uniqueness

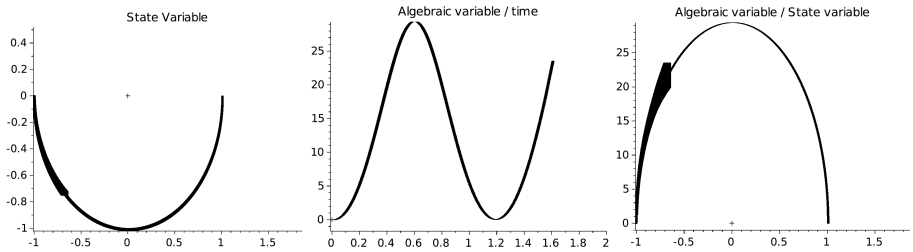


Figure 8: Plot of state variable p w.r.t. q (a), algebraic variable w.r.t. time (b) and algebraic variable w.r.t. state variable p (c) (for Problem 5.3) - with Algorithm 4: computation time 27 minutes

of solution, we develop a novel operator which simultaneously computes *a priori* enclosures of algebraic and state variables. For that, our operator mixes a classical Picard-Lindelöf (based on Taylor series) and a parametric Krawczyk operator. Then these enclosures are refined with a contractor programming approach by combining an integration scheme (Runge-Kutta RadauIIA) and an algebraic contractor (made with Krawczyk and forward/backward). Our complete algorithm is applied to three examples to show a proof of concept and the efficiency of the method. Moreover, a third contribution is also proposed in order to improve the result obtained. Indeed, it is often possible to obtain additive constraints on a physical system, and these constraints can be easily used as contractors for both state and algebraic variables. Finally, and to conclude the contributions, our approach is able to naturally verify the initial consistency of DAEs in index-1 Hessenberg form, which is one of the main issues in DAE community.

Several potential improvements to our method can be considered. The first one and the most important is the integration method used. Currently, we only tried the Radau IIA order 3 method. A serious improvement may be obtained with an higher order Runge-Kutta method such as Radau IIA order 5 (but its coefficients are not exact) or Gauss-Legendre order 6 (which has good properties but the computation of its local truncation error is time consuming). In addition, many improvements can be done on the global algorithm, such as a better step-size control and a better estimation for algebraic variables in the first trial of Picard-Krawczyk in order to avoid as much as possible the epsilon inflation. Finally, the first version of our tool DynIbex[2] used in this paper can be strongly improved to obtain a satisfying computation time. In order to illustrate the effect of tolerance on computation time and diameter of solution, Figure 9 shows computation time w.r.t. simulation time on Example 5.2 with 4 different tolerancies and a voluntary interruption of computation when diameter of solution is larger than 1. It is then obvious that the choice of a tolerance is strategic, depending on the goal.

References

[1] O.A. Akinfenwa and S.A Okunuga. Solving Semi-Explicit Index-1 DAE Systems using L-Stable Extended Block Backward Differentiation Formula with Continu-

[2] `http://perso.ensta-paristech.fr/~chapoutot/dynibex/`

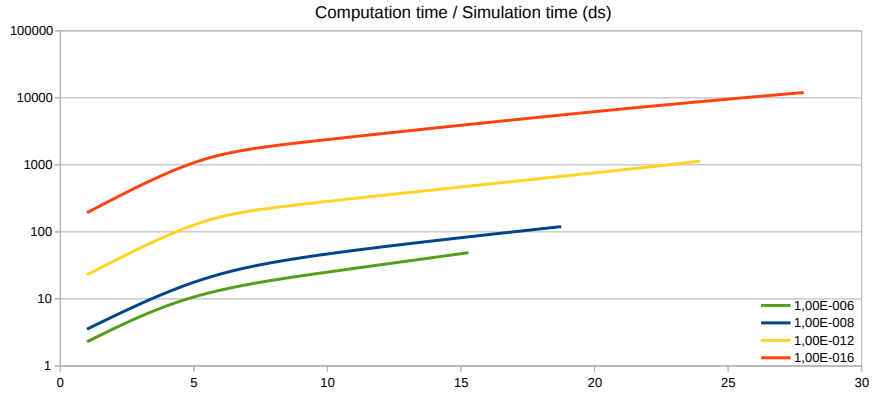


Figure 9: Computation time / simulation time (in tenths of a second) with 4 different tolerances and a voluntary interruption of computation if $w(y(t)) > 1$.

ous Coefficients. In *World Congress on Engineering*, pages 252–256, 2013.

[2] Julien Alexandre dit Sandretto and Alexandre Chapoutot. Validated solution of initial value problem for ordinary differential equations based on explicit and implicit Runge-Kutta schemes. Research report, ENSTA ParisTech, January 2015.

[3] M. Althoff and B.H. Krogh. Reachability analysis of nonlinear differential-algebraic systems. *Automatic Control, IEEE Transactions on*, 59(2):371–383, Feb 2014.

[4] F. Benhamou, F. Goualard, L. Granvilliers, and J.-F. Puget. Revising Hull and Box Consistency. In *Proc. ICLP*, pages 230–244, 1999.

[5] Martin Berz and Kyoko Makino. Verified integration of ODEs and flows using differential algebraic methods on high-order Taylor models. *Reliable Computing*, 4(4):361–369, 1998.

[6] Olivier Bouissou, Alexandre Chapoutot, and Adel Djoudi. Enclosing Temporal Evolution of Dynamical Systems Using Numerical Methods. In *NASA Formal Methods*, number 7871 in LNCS, pages 108–123. Springer, 2013.

[7] Olivier Bouissou and Matthieu Martel. GRKLib: A guaranteed Runge Kutta library. In *Scientific Computing, Computer Arithmetic and Validated Numerics*, 2006.

[8] Olivier Bouissou, Samuel Mimram, and Alexandre Chapoutot. HySon: Set-based simulation of hybrid systems. In *Rapid System Prototyping*. IEEE, 2012.

[9] David J. Braun and Michael Goldfarb. Simulation of constrained mechanical systems part I: An equation of motion. *Journal of Applied Mechanics*, 79(4), 2012.

[10] Kathryn Eleda Brenan, Stephen L Campbell, and Linda Ruth Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, volume 14. SIAM, 1996.

[11] John C. Butcher. Coefficients for the study of Runge-Kutta integration processes. *Journal of the Australian Mathematical Society*, 3:185–201, 5 1963.

[12] Gilles Chabert and Luc Jaulin. Contractor programming. *Artificial Intelligence*, 173(11):1079–1100, 2009.

[13] Xin Chen, Erika Abraham, and Sriram Sankaranarayanan. Taylor model flowpipe construction for non-linear hybrid systems. In *IEEE 33rd Real-Time Systems Symposium*, pages 183–192. IEEE Computer Society, 2012.

[14] L. H. de Figueiredo and J. Stolfi. *Self-Validated Numerical Methods and Applications*. Brazilian Mathematics Colloquium monographs. IMPA/CNPq, 1997.

[15] J Dieudonné. *Foundations of Modern Analysis*. Academic Press, 1969.

[16] P Eijgenraam. The solution of initial value problems using interval arithmetic. Technical report, Mathematical Centre, Tracts No.144, Stichting Mathematisch Centrum, Amsterdam, 1991.

[17] Karol Gajda, Malgorzata Jankowska, Andrzej Marciniak, and Barbara Szyszka. A survey of interval Runge–Kutta and multistep methods for solving the initial value problem. In *Parallel Processing and Applied Mathematics*, volume 4967 of *LNCS*, pages 1361–1371. Springer Berlin Heidelberg, 2008.

[18] A. Goldsztejn. Comparison of the Hansen-Sengupta and the Frommer-Lang-Schnurr existence tests. *Computing*, 79(1):53–60, 2007.

[19] A. Goldsztejn. Sensitivity analysis using a fixed point interval iteration. *arXiv preprint arXiv:0811.2984*, 2008.

[20] Ernst Hairer, Syvert Paul Norsett, and Grehard Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer-Verlag, 2nd edition, 2009.

[21] Jens Hoefkens, Martin Berz, and Kyoko Makino. Computing validated solutions of implicit differential equations. *Advances in Computational Mathematics*, 19(1-3):231–253, 2003.

[22] Tomas Kapela and Piotr Zgliczyǹski. A Lohner-type algorithm for control systems and ordinary differential inclusions. *Discrete and continuous dynamical systems - series B*, 11(2):365–385, 2009.

[23] Youdong Lin and Mark A. Stadtherr. Validated solutions of initial value problems for parametric ODEs. *Appl. Numer. Math.*, 57(10):1145–1162, 2007.

[24] Rudolf J. Lohner. Enclosing the solutions of ordinary initial and boundary value problems. *Computer Arithmetic*, pages 255–286, 1987.

[25] Ramon Moore. *Interval Analysis*. Prentice Hall, 1966.

[26] Ned Nedialkov, K. Jackson, and Georges Corliss. Validated solutions of initial value problems for ordinary differential equations. *Appl. Math. and Comp.*, 105(1):21 – 68, 1999.

[27] Nedialko (Ned) S. Nedialkov. *Computing Rigorous Bounds on the Solution of an Initial Value Problem for an Ordinary Differential Equation*. PhD thesis, University of Toronto, 1999.

[28] Arnold Neumaier. The wrapping effect, ellipsoid arithmetic, stability and confidence regions. In *Validation Numerics*, volume 9 of *Computing Supplementum*, pages 175–190. Springer Vienna, 1993.

[29] Constantinos C. Pantelides. The Consistent Initialization of Differential-Algebraic Systems. *SIAM Journal on Scientific and Statistical Computing*, 9(2):213–231, 1988.

[30] Andreas Rauh, Michael Brill, and Clemens GüNther. A novel interval arithmetic approach for solving differential-algebraic equations with ValEncIA-IVP. *International Journal on Applied Mathematical Computation Science*, 19(3):381–397, 2009.

[31] Robert Rihm. Interval methods for initial value problems in ODEs. *Topics in Validated Computations*, pages 173–207, 1994.

[32] Joseph K. Scott and Paul I. Barton. Interval Bounds on the Solutions of Semi-Explicit Index-one DAEs. Part 1: Analysis. *Numerische Mathematik*, 125(1):1–25, 2013.

[33] Joseph K. Scott and Paul I. Barton. Interval bounds on the solutions of semi-explicit index-one DAEs. Part 2: Computation. *Numerische Mathematik*, 125(1):27–60, 2013.

[34] N. F. Stewart. A heuristic to reduce the wrapping effect in the numerical solution of $x' = f(t, x)$. *BIT Numerical Mathematics*, 11(3):328–337, 1971.

[35] R. C. Vieira and E. C. Biscaia Jr. An overview of initialization approaches for differential algebraic equations. *Latin American Applied Research*, 30(4):303–313, 2000.

[36] Daniel Wilczak and Piotr Zgliczyński. Cr-Lohner algorithm. *Schedae Informaticae*, 20:9–46, 2011.