Validated Simulation of Differential Algebraic Equations with Runge-Kutta Methods

Julien Alexandre Dit Sandretto, Alexandre Chapoutot

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

Ensta ParisTech, Computer Science Department

Abstract

Differential Algebraic Equations (DAEs) are a general and implicit form of differential equations. This mathematical object is often used to represent physical systems such as dynamics of solid or chemical interactions. These equations are different from Ordinary Differential Equations (ODEs) in sense that some of the dependent variables occur without their derivatives. These variables are called "algebraic variables", which means free of derivatives and not with respect to abstract algebra.
Validated simulation of ODEs has recently known different developments such as guaranteed Runge-Kutta integration schemes, explicit and implicit ones. Not so far from an ODE, solving a DAE consists of searching 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

Chapter 1

Motivations

Our recent results on validated simulation of Ordinary Differential Equations (ODEs) with implicit Runge-Kutta schemes [2] lead us to go up in complexity of the kind of differential equations we want 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). Indeed, a DAE can be seen as a parametrized ODE. We currently focus on DAE in Hessenberg index-1 form, that is

$$\begin{cases} \dot{\mathbf{y}} = f(t, \mathbf{x}, \mathbf{y}) \\ 0 = g(t, \mathbf{x}, \mathbf{y}) \end{cases} \tag{1.1}$$

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

In Equation (1.1), $\mathbf{y} \in \mathbb{R}^n$ is the vector of state variables and $\mathbf{x} \in \mathbb{R}^m$ 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 as Simulink and Modelica-like software. In this paper, we will present a validated method to solve initial value problem given in the form of a DAE.

The article is organized as followed. 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 done. In Section 3, we will perform a review of literature about few existing validated approaches for DAEs. In Section 4, we present in details our method. In Section 5, we apply our solver on three different examples: a basic problem, a problem with exact solution known and the classic pendulum problem. In Section 6, we present how to take into account additional constraints in IVP for DAE before concluding in Section 7.

Notations

$\dot{y}$ denotes the time derivative of y, i.e., $\frac{dy}{dt}$. a denotes a real values while $\mathbf{a}$ represents a vector of real values. $[a]$ represents an interval values and $[\mathbf{a}]$ represents a vector of interval values. The midpoint of an interval $[x]$ is denoted by $\mathrm{m}([x])$. The variables y is assigned to the state variables of the system, when x is reserved for the algebraic variables and t obviously kept for the time. 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 variable. We also denote $y(t, y_j)$

the exact solution of a differential equation at time t with a known value y_j at time t_j, in our case it implies that $t > t_j$. Sets will be represented by calligraphic letter such as $\mathcal{X}$ or $\mathcal{Y}$.

Chapter 2

Validated simulation of Ordinary differential equations

A simulation of an ordinary differential equation consists on a discretization of time and an iterative and approximative computation of the state of the system, by the help of an integration scheme. In details, an integration scheme, starting from an initial value $\mathbf{y}_n$ at time t_n and a finite time horizon h, the stepsize, produces an approximation $\mathbf{y}_{n+1}$ at time t_{n+1}, with $t_{n+1} - t_n = h$, of the solution $\mathbf{y}(t_{n+1}; \mathbf{y}_n)$. In this section, we briefly recall how a validated simulation of an ODE is computed. We refer to [26] for a more detailled presentation. The notations follow [26].

2.1 Ordinary differential equations

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

$$\dot{\mathbf{y}} = f(t, \mathbf{y}) \quad \text{with} \quad \mathbf{y}(0) = \mathbf{y}_0,\ \mathbf{y}_0 \in \mathbb{R}^n \quad \text{and} \quad t \in [0, t_{\text{end}}]\ . \tag{2.1}$$

In the classical approach [24, 26] to define validated method for IVP, each step of an integration scheme consists in two steps: *a priori enclosure* and *solution tightening*. Starting from a valid enclosure $[\mathbf{y}_j]$ at time t_j, the two following 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 $h_j = t_{j+1} - t_j$ is as larger as possible in terms of accuracy and existence proof for the IVP solution.

Step 2. Compute a tighter enclosure of $[\mathbf{y}_{j+1}]$ 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

[25, 24, 26]. This phenomenon appears when we try to enclose a set with an interval vector (geometrically a box). The arising 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 2.1.

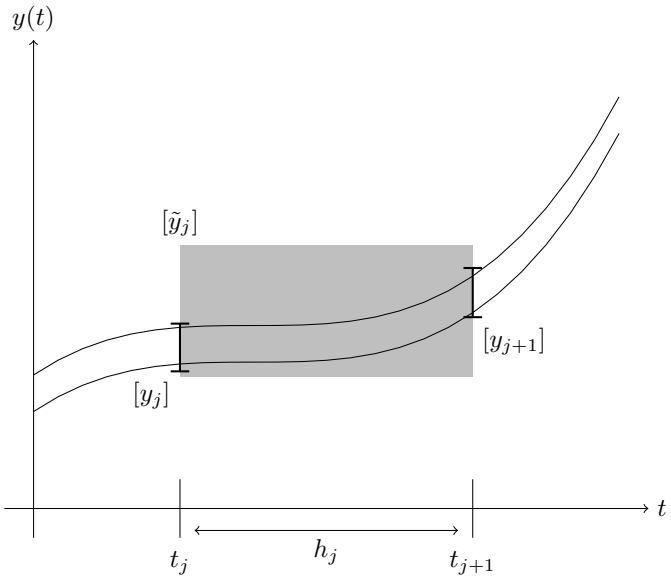


Figure 2.1: Enclosures appeared during one step

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

2.2 A priori solution enclosure

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

Theorem 2.2.1 (Banach fixed-point theorem) *Let (K, d) a complete metric space and let $g : K \to K$ 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 context of IVP, we consider the space of continuously differentiable functions $C^0([t_j, t_{j+1}], \mathbb{R}^n)$ and the Picard-Lindelöf operator

$$P_f(y) = t \mapsto \mathbf{y}_j + \int_{t_n}^{t} f(s, \mathbf{y}(s))ds \ . \tag{2.2}$$

Note that this operator is associated to the integral form of Equation (2.1). So the solution of this operator is also the solution of Equation (2.1).

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 (2.1) 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

$$P_f([\mathbf{r}]) = [\mathbf{y}_j] + [0, h] \cdot f([\mathbf{r}]), \tag{2.3}$$

with $h = t_{j+1} - t_j$ the step-size. Theorem 2.2.1 says that if we can find $[\mathbf{r}]$ such that $P_f([\mathbf{r}]) \subseteq [\mathbf{r}]$ then the operator is contracting and Equation (2.1) 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 (2.1).

Remark that the operator defined in Equation (2.3) can also define a contractor (in a sens of interval analysis [12]) on $[\mathbf{r}]$ after the contraction proved for P_f such that

$$[\mathbf{r}] \leftarrow [\mathbf{r}] \cap [\mathbf{y}_j] + [0, h].f([\mathbf{r}]) \ . \tag{2.4}$$

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 (2.3) and its associated contractor defined in Equation (2.4) can be defined over a more accurate integration scheme (at the 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

Interval version of Taylor series for ODE integration gives

$$[\mathbf{y}_{j+1}] \subset \sum_{k=0}^{N} f^{[k]}([\mathbf{y}_j])h^k + f^{[N+1]}([\tilde{\mathbf{y}}_j])h^{N+1}, \tag{2.5}$$

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

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

$$\mathbf{y}_{j+1}([t_j, t_{j+1}]; [\mathbf{r}]) = \mathbf{y}_j + \sum_{k=0}^{N} f^{[k]}([\mathbf{y}_j])[0, h^k] + f^{[N+1]}([\mathbf{r}])[0, h^{N+1}] \ . \tag{2.6}$$

In consequence, if $[\mathbf{r}] \supseteq \mathbf{y}_{j+1}([t_j, t_{j+1}], [\mathbf{r}])$, then Equation (2.6) defined a contraction map and Theorem 2.2.1 can be applied.

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

Note that the scheme defined in Equation (2.5) is usually evaluated in a centered form for a more accurate result

$$[\mathbf{y}_{j+1}] \subset \sum_{k=0}^{N} f^{[k]}(\hat{\mathbf{y}}_j)h^k + f^{[N+1]}([\tilde{\mathbf{y}}_j])h^{N+1} + \left(\sum_{k=0}^{N} J(f^{[k]}, [\mathbf{y}_j])h^i)([\mathbf{y}_j] - \hat{\mathbf{y}}_j\right), \quad (2.7)$$

with $\hat{\mathbf{y}}_j \in [\mathbf{y}_j]$ $J(f^{[k]}, [\mathbf{y}_j])$ 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 wrapping effect [26]. These two "tricks", with a strong computational cost, can be avoided by using the affine arithmetic.

Picard-Lindelöf operator, as defined in Equation (2.6), gives an *a priori* enclosure $[\mathbf{r}]$, using Theorem 2.2.1. Picard-Lindelöf operator is proven to be contracting on $[\mathbf{r}]$, we can then use this operator to contract the box $[\mathbf{r}]$ till 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 very important to contract as much as possible this box $[\mathbf{r}]$ 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 we dispose of the 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 in 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 $LTE_\Phi(t, t_j, [\mathbf{y}_j])$. Indeed, a guaranteed integration scheme has the property that there exists a time $\xi \in [t_j, t_{j+1}]$ such that

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

So $[\mathbf{y}_{j+1}] = \Phi(t_{j+1}, t_j, [\mathbf{y}_j]) + LTE_\Phi(\xi, t_j, [\mathbf{y}_j])$ 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 [25, 24, 5, 26, 22, 36, 13, 23],
2. Interval Runge-Kutta methods [17, 7, 6].

The former is the oldest method used in this context. Indeed, R. Moore [25] already applied this method in the sixties and until now it is the most used method to solve Equation (2.1). The latter is more recent, see in particular [7, 6], but Runge-Kutta methods have many interesting properties, 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 known as the number of *stages* of a Runge-Kutta method. More precisely, a Runge-Kutta method is defined by

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h \sum_{i=1}^{s} b_i \mathbf{k}_i \ , \tag{2.8}$$

with $\mathbf{k}_i$ defined by

$$\mathbf{k}_i = f\left(t_n + c_i h, \mathbf{y}_n + h \sum_{j=1}^{s} a_{ij} \mathbf{k}_j \right) \ . \tag{2.9}$$

The coefficient c_i, a_{ij} and b_i, for $i, j = 1, 2, \cdots, s$, fully characterize the Runge-Kutta methods and their are usually synthesized in a *Butcher tableau* [11] of the form

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

In function of the form of the matrix A, made of the coefficients a_{ij}, a Runge-Kutta method can be

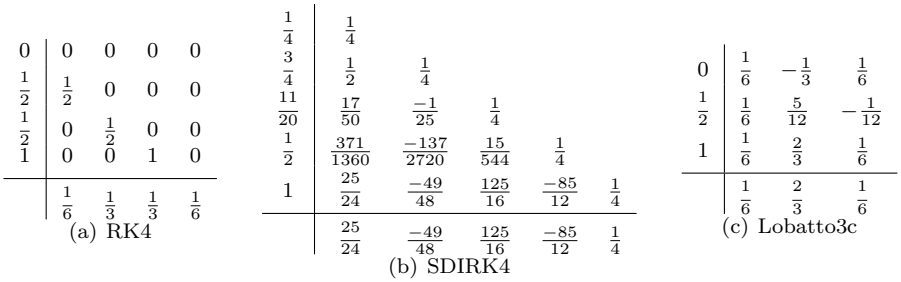
- *explicit*, e.g., the classical Runge-Kutta method of order 4 given in Figure 2.2(a). In other words, the computation of an intermediate $\mathbf{k}_i$ only depends on the previous steps $\mathbf{k}_j$ for $j < i$. These methods are often the faster for a given order, but with less properties than the followers;
- *diagonally implicit*, e.g., a diagonally implicit method of order 4 given in Figure 2.2(b). In this case, the computation of an intermediate step $\mathbf{k}_i$ involves the value $\mathbf{k}_i$ and so non-linear systems in $\mathbf{k}_i$ must be solved. Nevertheless, the structure of this kind of method leads to a faster solving;
- *fully implicit*, e.g., the Runge-Kutta method with a Lobatto quadrature formula of order 4 given in Figure 2.2(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.

Local truncature error

Recently, we defined an unified approach to express LTE for explicit and implicit Runge-Kutta methods. More precisely, for a Runge-Kutta of order p we have

$$\begin{aligned} \mathrm{LTE}(t, \mathbf{y}(\xi)) := y(t_n; \mathbf{y}_{n-1}) - \mathbf{y}_n = \\ \frac{h^{p+1}}{(p+1)!} \sum_{r(\tau)=p+1} \alpha(\tau) \left[1 - \gamma(\tau)\psi(\tau) \right] F(\tau)(\mathbf{y}(\xi)) \quad \xi \in [t_n, t_{n+1}] \end{aligned} \tag{2.10}$$

with



(a) RK4

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

(b) SDIRK4

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

(c) Lobatto3c

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

Figure 2.2: Different kinds of Runge-Kutta methods

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

Note that $\mathbf{y}(\xi)$ is a particular, and *a priori* unknown, solution of Equation (2.1) at a time instant ξ. This solution can be over-approximated using Picard-Lindelöf operator presented in Section 2.2.

Wrapping effect

Runge-Kutta methods also suffer from the wrapping effect, such 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]. An other way is to modify the geometry of the enclosing set (parallelepipeds [16, 25], ellipsoids [28], convex polygons [31] and zonotopes [34, 8]).

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

Chapter 3

Differential Algebraic Equations

We can distinguish at least two families of DAEs, the fully implicit ones and the semi-explicit ones, the fully explicit are similar to ODEs.

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 such as:

$$f(t, \mathbf{y}, \dot{\mathbf{y}}, ...) = 0, \quad t_0 \leq t \leq t_{end} \tag{3.1}$$

The order of this kind of DAE is defined with respect to the highest order of present derivatives. For example, if f is function of $\dot{\mathbf{y}}$ and no higher derivatives then f is a 1^{st} order DAE. If $\ddot{\mathbf{y}}$ occurs then f is a 2^{nd} order DAE, and so on. Nevertheless, all DAEs can be rewritten in DAE of 1^{st} order by increasing the size of the system.

3.2 Semi-explicit DAEs or in Hessenberg form

The second, and the most used DAE form in science and engineering, is the semi-explicit DAEs, or also called DAEs in Hessenberg form. In this formalism, the index is a differentiation index [10], that is to say the distance to the related ODE. For example, the index 1 Hessenberg form is described by:

$$\begin{cases} \dot{\mathbf{y}} = f(t, \mathbf{x}, \mathbf{y}) \\ 0 = g(t, \mathbf{x}, \mathbf{y}) \end{cases} \tag{3.2}$$

while the index 2 Hessenberg form is described by:

$$\begin{cases} \dot{\mathbf{y}} = f(t, \mathbf{x}, \mathbf{y}) \\ 0 = g(t, \mathbf{x}) \end{cases} \tag{3.3}$$

In the way that some of dependent variables occur without their derivatives, these differential systems are different from an ODE with an additive constraint (which are independent).

3.3 Review of literature

If there exists many tools for solving DAEs in a numerical way (DAETS, Sundials, Mathematica) and some particular implementation embedded in simulation softwares (Simulink, Dymola), only few attempts have been done in a guaranteed approach.

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 to compute 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 till validating an inclusion test. The approach presented is then different than our method. Unfortunately, none of these two latter approaches seems to be continued.

Finally, and more recently, one important work has been done in [32, 33]. In these papers, 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 real analysis and comparison. The results presented are also difficult to estimate. Nevertheless, we are able to make few remarks. Firstly, the presented approach is not completely validated (as honestly said in the paper). Secondly, if it is well known that Hansen-Sengupta test is more efficient than Krawczyk one [18], in [32] it is used in non-parametric version while we advocate to use Krawczyk in parametric version. Indeed, Hansen-Sengupta requires the solving of a linear interval system, which can be time consuming function of problem size and done many times at each integration step (till obtain conditions of Theorem 2.2.1). Moreover, the operator is applied to a non linear function with state variable as interval parameter, which can be large due to initial state, 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 issue [35].

Chapter 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:

$$\begin{cases} \dot{\mathbf{y}} = f(t, \mathbf{x}, \mathbf{y}) \\ 0 = g(t, \mathbf{x}, \mathbf{y}) \end{cases} \quad \text{with} \quad \mathbf{y}(0) \in [\mathbf{y}_0] \quad \text{and} \quad \mathbf{x}(0) \in [\mathbf{x}_0] \ . \tag{4.1}$$

In Section 2.1, we recalled the classical two steps method presented before 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 solution of the DAE on an integration step $[t, t+h]$ with a novel Picard-like operator;
- Refine this enclosure at $t+h$ with integration Runge-Kutta scheme and contractors.

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

4.1 New Picard-like operator

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

$$\dot{\mathbf{y}} = f(t, \mathbf{y}, \mathbf{p}) \quad \text{with} \quad \mathbf{y}(0) \in [\mathbf{y}_0] \quad \text{and} \quad \mathbf{p} \in [\mathbf{p}] \ . \tag{4.2}$$

We have the habit to use the Picard-Lindelöf operator. The Picard-Lindelöf operator, based on the Theorem 2.2.1 and defined in Equation (2.2), allows one to compute the *a priori* enclosure $[\tilde{\mathbf{y}}_j]$ such that

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

In the case of a DAE expressed by Equation (4.1), 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 reversal, 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 and uncomputable. Anyway, this approach leads us to the following solution.

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

Theorem 4.1.1 (Part of Frobenius theorem) *Let $\mathcal{X}$ and $\mathcal{Y}$ be Banach spaces, and $\mathcal{A} \subset \mathcal{X}$, $\mathcal{B} \subset \mathcal{Y}$ a pair of 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 X into Y. A differentiable mapping $u : \mathcal{A} \to \mathcal{B}$ is a solution of the differential equation*

$$\dot{y} = F(x, y) \tag{4.3}$$

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

Equation (4.3) is completely integrable if for each $(x_0, y_0) \in A \times B$, there is a neighborhood $\mathcal{U}$ of x0 such that Equation (4.3) has a unique solution $u(x)$ defined on $\mathcal{U}$ such that $u(x_0) = y_0$.

Theorem 4.1.2 (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 4.1.1 (Set membership view of Frobenius) *Let $\mathcal{A}$ and $\mathcal{B}$ two Banach spaces and g, f two continuously differentiable function of 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:*

$$\dot{y} = f(t, x, y) \tag{4.4}$$

and

$$g(x, y) = 0 \tag{4.5}$$

Let $\mathcal{X} \subset \mathcal{A}$, and $\mathcal{Y} \subset \mathcal{B}$ be a pairs of set. If $\forall y \in \mathcal{Y}, \exists!\ x \in \mathcal{X}$ satisfaying Equation (4.5) and $\forall x \in \mathcal{X}, \exists!\ y \in \mathcal{Y}$ satisfaying Equation (4.4), simultaneously, then Equation (4.4) 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 prove that there is a unique function $x = \psi(y)$ for all $(x, y) \in \mathcal{A} \times \mathcal{B}$. Then the differential Equation (4.4) becomes $\dot{y} = f(t, \psi(y), y)$ with $\psi(y)$ 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 (4.4) in $\mathcal{Y}$. □

4.1.2 Adapted to DAEs

A direct translation of Proposition 4.1.1 to our problem gives:

Proposition 4.1.2 *Let* $[\tilde{\mathbf{y}}]$ *be a Picard solution of* $\dot{\mathbf{y}} \in f(t, [\tilde{\mathbf{x}}], \mathbf{y})$ *and* $[\tilde{\mathbf{x}}]$ *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 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 4.1.1. □

4.1.3 The operator

Finally, with the last Proposition 4.1.2, we define a novel operator Picard-Krawczyk $\mathcal{PK}$:

$$\text{If } \begin{pmatrix} \mathcal{P}([\tilde{\mathbf{y}}], [\tilde{\mathbf{x}}]) \\ \mathcal{K}([\tilde{\mathbf{y}}], [\tilde{\mathbf{x}}]) \end{pmatrix} \subset Int \begin{pmatrix} [\tilde{\mathbf{y}}] \\ [\tilde{\mathbf{x}}] \end{pmatrix} \quad \text{then} \quad \exists! \quad \text{solution of DAE in } [\tilde{\mathbf{y}}] \tag{4.6}$$

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 details, the operators are given by:

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

$$\mathcal{P}_f([\mathbf{y}_j], [\mathbf{x}_j], [\mathbf{r}], [\tilde{\mathbf{x}}], h) = [\mathbf{y}_j] + \sum_{k=0}^{N} f^{[k]}([\mathbf{x}_j], [\mathbf{y}_j])[0, h^k] + f^{[N+1]}([\tilde{\mathbf{x}}], [\mathbf{r}])[0, h^{N+1}] \ . \tag{4.7}$$

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

Parametric preconditioned Krawczyk operator:

$$\begin{aligned} \mathcal{K}_g([\tilde{\mathbf{y}}], [\mathbf{r}]) = \mathrm{m}([\mathbf{r}]) - Cg(\mathrm{m}([\mathbf{r}]), \mathrm{m}([\tilde{\mathbf{y}}])) - \\ (C\frac{\partial g}{\partial x}([\mathbf{r}], [\tilde{\mathbf{y}}]) - I)([\mathbf{r}] - \mathrm{m}([\mathbf{r}])) - \\ C\frac{\partial g}{\partial y}(\mathrm{m}([\mathbf{r}]), [\tilde{\mathbf{y}}])([\tilde{\mathbf{y}}] - \mathrm{m}([\tilde{\mathbf{y}}])) \end{aligned} \tag{4.8}$$

If $\mathcal{K}_g([\tilde{\mathbf{y}}], [\mathbf{r}]) \subset Int([\mathbf{r}])$ 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:

An 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 $\mathcal{K}_g([\tilde{\mathbf{y}}], [\mathbf{r}]) \subset Int([\mathbf{r}])$ 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 ${}^0[\tilde{x}]$, ${}^0[\tilde{y}]$, a tolerancy on LTE tol and a guessed step size h. The outputs are wheither a success flag with the validated enclosures ${}^1[\tilde{x}]$, ${}^1[\tilde{y}]$ and the already computed lte, either a fail flag. We iterate two times the size of function f in order to prapagate enoughly the inflation.

Algorithm 1 Compute the *a priori* enclosures

Require: ${}^0[\tilde{\mathbf{x}}]$, ${}^0[\tilde{\mathbf{y}}]$, tol, h, $iter = 0$
  ${}^1[\tilde{\mathbf{x}}] = \mathcal{K}({}^0[\tilde{\mathbf{y}}], {}^0[\tilde{\mathbf{x}}])$ // 4.1.3
  ${}^1[\tilde{\mathbf{y}}] = \mathcal{P}({}^0[\tilde{\mathbf{y}}], {}^0[\tilde{\mathbf{x}}])$ // 4.1.3
  while $({}^1[\tilde{\mathbf{x}}] \not\subset {}^0[\tilde{\mathbf{x}}])$ **and** $({}^1[\tilde{\mathbf{y}}] \not\subset {}^0[\tilde{\mathbf{y}}])$ **and** $(iter < 2 * size(f))$ **do**
    $iter = iter + 1$
    ${}^0[\tilde{\mathbf{y}}] = {}^1[\tilde{\mathbf{y}}] \pm 1\%$
    ${}^0[\tilde{\mathbf{x}}] = {}^1[\tilde{\mathbf{x}}] \pm 1\%$
    ${}^1[\tilde{\mathbf{x}}] = \mathcal{K}({}^0[\tilde{\mathbf{y}}], {}^0[\tilde{\mathbf{x}}])$ // 4.1.3
    ${}^1[\tilde{\mathbf{y}}] = \mathcal{P}({}^0[\tilde{\mathbf{y}}], {}^0[\tilde{\mathbf{x}}])$ // 4.1.3
  end while
  if $({}^1[\tilde{\mathbf{x}}] \subset {}^0[\tilde{\mathbf{x}}])$ and $({}^1[\tilde{\mathbf{y}}] \subset {}^0[\tilde{\mathbf{y}}])$ **then** // conditions of 4.1.2 obtained
    lte = LTE(${}^1[\tilde{\mathbf{x}}]$, ${}^1[\tilde{\mathbf{y}}]$) // 2.4
    if lte $<$ tol **then** // Acceptable lte
      return SUCCESS(${}^1[\tilde{\mathbf{x}}]$, ${}^1[\tilde{\mathbf{y}}]$, lte)
    end if
  end if
  return FAILED // No enclosures found with the inputs

4.2 Contractors

After the guaranteed enclosures on $[t, t+h]$ obtained, we can contract these enclosures around $t + h$, because obviously $\mathbf{x}(t + h) \subset [\tilde{\mathbf{x}}]$ and $\mathbf{y}(t + h) \subset [\tilde{\mathbf{y}}]$. Firstly, 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 differential equation is often stiff (this assumption will be verified in section 5.1) and has interval coefficient. 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 method at order 3 for stiff problem. Its Butcher tableau is given in Fig. 4.1.

This IRK method attains an order three with two stages, it is fully implicit as shown in its tableau (Fig. 4.1) and A-stable. The corresponding local truncature error (lte) can be computed by using the Butcher trees as shown in [2].

Secondly, 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

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

Figure 4.1: Butcher Tableau of Radau IIA order 3 (an Implicit Runge-Kutta method)

operator from Section 4.1 and a forward/backward contractor coming from constraint programming. The Forward/Backward contractor (also called HC4-Revise [4]) 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 our tool.

Finally, these two obtained contractors are embedded in a fixpoint presented in Algorithm 2. The inputs of this algotithm 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
    $[\mathbf{y}_{j+1}] \cap = (RADAU3_f([\mathbf{x}_{j+1}], [\mathbf{y}_{j+1}]) + lte)$ // 2.4 and 4.1
    $[\mathbf{x}_{j+1}] \cap = (FwdBwd_g([\mathbf{x}_{j+1}], [\mathbf{y}_{j+1}]) \cap \mathcal{K}_g([\mathbf{x}_{j+1}], [\mathbf{y}_{j+1}]))$ // [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, a given minimal step size and a tolerancy. The algorithm builds a list of data computed at each integration step, as described in Section 2.1.

Algorithm 3 Simulation of DAE

Require: $y(0)$, $x(0)$, T_{final}, h_{min}, tol // Problem statement
  $t = 0$
  while $(t < T_{final})$ **do** // Till desired time
    $[\tilde{x}] = x(t), [\tilde{y}] = y(t)$
    Computation of $[\tilde{x}], [\tilde{y}], lte$ // Alg. 1
    if SUCCESS **then**
      Computation of $[x(t+h)], [y(t+h)]$ // Alg. 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

Chapter 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 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:

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

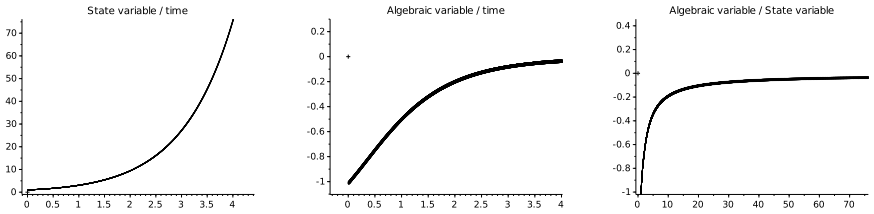
We perform a simulation till $T_{final} = 4$ seconds with a desired tolerancy $tol = 1e-16$. The initial value consistency is checked with Krawczyk which leads to $x(0) = -1$. The computation takes between 16 and 30 seconds depending of computer. Our tool provides three files containing:

- 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 Tab. 5.1

With the files containing state and algebraic values w.r.t. time, we are able to plot three figures, given in Fig. 5.1. On the figure (a) and (b), it is apparent that even if state variable and algebraic variable evolves exponentially but quite slowly, algebraic variable evolves in a stiff way w.r.t. state variable (c). In general, DAEs leads to a stiff problem, which is the motivation for using RADAU IIA method. This method is known for its efficiency in front of stiff problems.

Log.txt file	Description
`Solution at t=4.00000 :`	Final time reached
`([76.2255, 76.2294])`	Solution for state variable
`Diameter :  (0.00395156)`	Diameter of the solution
`Rejected Picard :11496`	Number of step rejected by Alg. 1
`Accepted Picard :21743`	Number of step accepted by Alg. 1
`Step min :2.03865e-05`	Time step minimum (h)
`Step max :0.00025`	Time step maximum (h)
`Truncature error max :2.7141e-15`	Maximum lte during simulation

Table 5.1: Log file and its description



(a) (b) (c)

Figure 5.1: 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 prob. 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:

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

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

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 till $T_{final} = 2$ seconds with a desired tolerancy $tol = 1e-22$. The initial value consistency is checked with Krawczyk which leads to

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

The results at final time is given in Tab. 5.2. These results allow us to verify the enclosure of 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 simulation. Moreover, at $t = 2$ seconds, the diameter of our solution is lower than 0.00056 when one of the best numerical method (without guarantee) [1], Extended Block Backward Differentiation Formula (EBBDF) at order 4, provides a result at 0.0002 from the exact value, which is comparable to our results (see Tab. 5.2).

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 5.2: Solution at $t = 2$ with our method, exact value computation and EBBDF results for prob. 5.2

5.3 Classical problem: Pendulum

Finally, the last example is the well known pendulum, expressed in index 1 Hessenberg form. The problem statement is defined in Fig. 5.2.

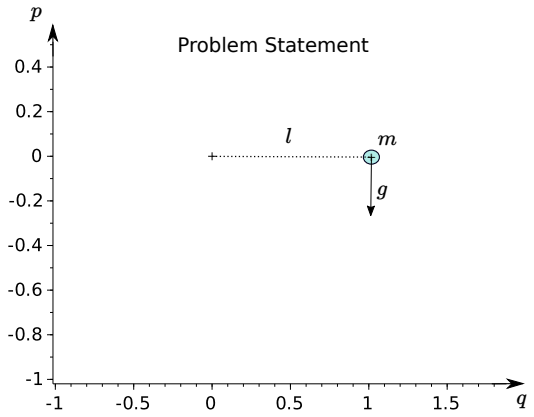


Figure 5.2: Problem statement of Ex. 5.3

A ball of mass m, suspended to a massless rod of length l, under gravity g is defined by coordinates (p, q) such that:

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

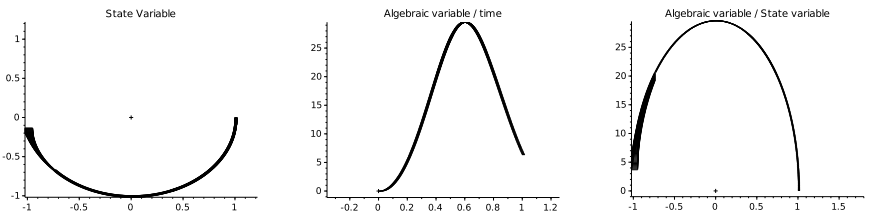
with u and v the velocity of pendulum, and λ the force in the rod. And the algebraic relation:

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

We perform a simulation till $T_{final} = 1$ second with a desired tolerancy $tol = 1e - 10$. 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 Fig. 5.3.

5.4 Discussion

We see through these three examples that our method is correct (Ex. 5.2), efficient in comparison to the existing non guaranteed methods (Ex. 5.2), able to solve a classical problem (Ex. 5.3), while providing statistics on the computation progress (Ex. 5.1). Nevertheless, we can also conclude that the diameter of solution growth 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.



(a) (b) (c)

Figure 5.3: 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 prob. 5.3)

Chapter 6

Additive contractors

Our tool is based on contractor programming [12], it means that we are able to add any contractors to the Alg. 2 to take into account some other constraints than $g(t, x, y) = 0$. In the case of physical systems, constraints can appear from the context such as energy conservation, mechanical constraints, or whatever. This approach is substantially similar to [9]. In this latter, authors use the energy conservation as constraint in order to improve the precision of a numerical simulation. More interesting, some constraints come directly from the Pantelides algorithm [29] which is used to reduce the order of DAEs.

For example, on the case of Pendulum (Ex. 5.3), Pantelides algorithm is used to obtained $m(u^2+v^2)-gq-l^2\lambda = 0$ by differentiation of circle equation $p^2+q^2-l^2 = 0$:

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

These three additives constraints can be used to generate a forward/backward contractor (with a propagation principle between constraints) for both state and algebraic variables. This contractor improve the result of our algorithm by changing Alg. 2 to Alg. 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**
    $[y_{j+1}]\cap = (RADAU3_f([x_{j+1}], [y_{j+1}]) + lte)$
    $[x_{j+1}]\cap = (FwdBwd_g([x_{j+1}], [y_{j+1}]) \cap \mathcal{K}_g([x_{j+1}], [y_{j+1}]))$
    $([x_{j+1}]; [y_{j+1}])\cap = FwdBwd_{ctc}(([x_{j+1}]; [y_{j+1}]))$
  end while

Our tool is applied with Alg. 2 and Alg. 4 to the Pendulum problem for a simulation till $T_{final} = 1.6$ seconds and for a desired tolerancy $tol = 10^{-18}$. The results are shown on Fig. 6.1 and Fig. 6.2. 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.

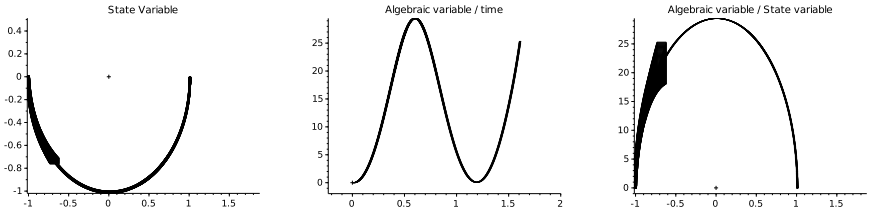


Figure 6.1: Plot of state variable p w.r.t. q (a), algebraic variable w.r.t. time (b) and algebraic variable w.r.t. state variable p (c) (for prob. 5.3) - with Alg. 2: computation time 28 minutes

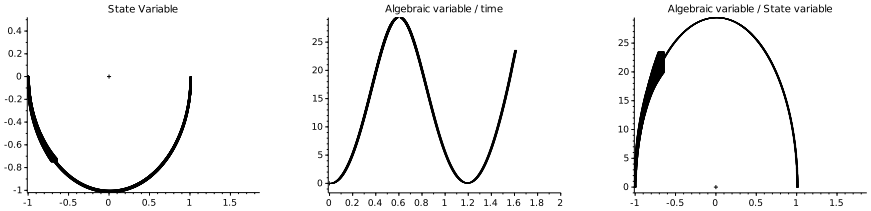


Figure 6.2: Plot of state variable p w.r.t. q (a), algebraic variable w.r.t. time (b) and algebraic variable w.r.t. state variable p (c) (for prob. 5.3) - with Alg. 4: computation time 27 minutes

Chapter 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 unicity of solution, we develop a novel operator which compute simultaneously the *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 is 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 that we thought sufficient to prove the 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 on the contributions, our approach can naturally solve the initial consistency, which is one of the main issue 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 have tried only Radau IIA order 3 method. A serious improvement can be probably obtained with an higher order Runge-Kutta method such as Radau IIA order 5 (but its coefficients are not exact) or Gauss order 6 (which has good properties but its local truncature error is computation time consuming). In addition, many improvement can be done on the global algorithm such as a better stepsize control and a better estimation for algebraic variable in the first trial of Picard-Krawczyk in order to reduce the number of rejected steps that is very important for the computation time.

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

Bibliography

[1] O.A. Akinfenwa and S.A Okunuga. Solving semi-explicit index-1 dae systems using l-stable extended block backward differentiation formula with continuous 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, Ma*ł*gorzata 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] Alexandre Goldsztejn. Sensitivity analysis using a fixed point interval iteration, 811.

[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] JosephK. Scott and PaulI. Barton. Interval bounds on the solutions of semi-explicit index-one daes. part 1: analysis. *Numerische Mathematik*, 125(1):1–25, 2013.

[33] JosephK. Scott and PaulI. Barton. Interval bounds on the solutions of semi-explicit index-one daes. part 2: computation. *Numerische Mathematik*, 125(1):27–60, 2013.

[34] NF Stewart. A heuristic to reduce the wrapping effect in the numerical solution ofx= 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.