SCHEDAE INFORMATICAE
VOLUME 20 (2011) DOI 10.4467/20838476SI.11.001.0287

$\mathcal{C}^r$-Lohner Algorithm[1]

Daniel Wilczak, Piotr Zgliczyński
Institute of Computer Science, Jagiellonian University,
Łojasiewicza 6, 30–348 Cracow, Poland
e-mail: *wilczak@ii.uj.edu.pl, umzglicz@cyf-kr.edu.pl*

Abstract. We present a Lohner type algorithm for the computation of rigorous bounds for the solutions of ordinary differential equations and its derivatives with respect to the initial conditions up to an arbitrary order.

Keywords: rigorous integration of ODEs, variational equations.

1. Introduction

This paper is a sequel to [31]. We present here a Lohner-type algorithm for the computation of rigorous enclosures of the partial derivatives with respect to the initial conditions up to an arbitrary order r of the flow induced by an autonomous ODE, hence the name "$\mathcal{C}^r$-Lohner algorithm". Let r be a positive integer. By the $\mathcal{C}^r$-algorithm we will mean a routine which gives rigorous estimates for the partial derivatives with respect to the initial conditions up to the order r. By the $\mathcal{C}^r$-computations we mean an application of the $\mathcal{C}^r$-algorithm.

Our main motivation for the development of the $\mathcal{C}^r$-algorithm was to provide a tool, which will considerably extend the possibilities of computer assisted proofs in the dynamics of ODEs requiring rigorous bounds on orbits. Till now, most of such proofs have used the topological conditions (see for example [10, 16, 6, 30]) and additionally the conditions on the first derivatives with respect to the initial conditions (see for example [21, 23, 27, 11]), hence they required the $\mathcal{C}^0$- and $\mathcal{C}^1$-computations, respectively. The spectrum of problems addressed includes the questions of the

[1]Research supported by an annual national scholarship for young scientists from the Foundation for Polish Science and, in part, by Polish State Ministry of Science and Information Technology grant N201 024 31/2163.

existence of periodic orbits and their local uniqueness, the existence of symbolic dynamics, the existence of hyperbolic invariants sets, the existence of homo- and heteroclinic orbits. To address other phenomena, like the bifurcations of periodic orbits, the route to chaos, invariant tori through the KAM theory, one needs the knowledge of partial derivatives with respect to the initial conditions of the higher order.

In principle, one can think that a good rigorous ODE solver should be enough. Namely, to compute the partial derivatives of the flow induced by

$$x' = f(x), \qquad x \in \mathbb{R}^n \tag{1}$$

it is enough to rigorously integrate the system of variational equations obtained by the formal differentiation of (1) with respect to the initial conditions. For example, for $r = 2$ we have the following system:

$$\begin{aligned} x' &= f(x), & (2)\\ \frac{d}{dt}V_{ij}(t) &= \sum_{s=1}^{n} \frac{\partial f_i}{\partial x_s}(x)V_{sj}(t), & (3)\\ \frac{d}{dt}H_{ijk}(t) &= \sum_{s,r=1}^{n} \frac{\partial^2 f_i}{\partial x_s \partial x_r}(x)V_{rk}(t)V_{sj}(t) + \sum_{s=1}^{n} \frac{\partial f_i}{\partial x_s}(x)H_{sjk}(x), & (4) \end{aligned}$$

with the initial conditions

$$x(0) = x_0, \quad V(0) = \mathrm{Id}, \qquad H_{ijk}(0) = 0, \quad i,j,k = 1,\ldots,n. \tag{5}$$

It is well known that if by $\varphi(t, x_0)$ we denote the (local) flow induced by (1), then

$$\begin{aligned} \frac{\partial \varphi_i}{\partial x_j}(t, x_0) &= V_{i,j}(t),\\ \frac{\partial^2 \varphi_i}{\partial x_j \partial x_k}(t, x_0) &= H_{ijk}(t). \end{aligned}$$

Analogous statements are true for the higher order partial derivatives with respect to the initial conditions.

Remark 1. *The variational equations up to an arbitrary order might be generated automatically by means of the automatic differentiation [5, 20]. The main reason for which we discuss in this paper an explicit compact formula for the equations of variations (see Eq. (3, 4) and Section 2) is to explain a method for the generation of the rough enclosure for the solution of higher order variational equations. In the practical implementation the use of any compact formulas for variational equations can be avoided.*

It turns out that a straightforward application of any rigorous ODE solver to the system of variational equations (2–4) is very inefficient. Namely, it totally ignores the structure of the system and sometimes it leads to a very poor performance and unnecessary long computation times (see Section 3.1 for more discussion and Section 7 for results of our tests).

Our algorithm is a modification of the Lohner algorithm [13], which takes into account the structure of variational equations (2–4). Basically, it consists of the Taylor method, a heuristic routine for a priori bounds for the solution of (2–4) during a time step and a Lohner-type control of the wrapping effect, which is done separately for x and the partial derivatives with respect to the initial conditions (the variables V and H in (3,4)). The proposed algorithm has been already successfully applied to several problems. In [12] a computer assisted proof of the existence of the cocooning cascade of heteroclinic tangencies for the Michelson system [14] was given. This proof required the $\mathcal{C}^2$ computations. That time we had a special implementation of the $\mathcal{C}^2$ algorithm only.

In [28] the method for proving the existence of quadratic homoclinic tangencies for maps is proposed. An application of the method to a Poincaré map for the forced-damped pendulum system required $\mathcal{C}^2$ computations. In [29] an application of the $\mathcal{C}^3$ algorithm to rigorous verification of period doubling bifurcations for the Rössler system [22] is presented.

In [25] $\mathcal{C}^3$ and $\mathcal{C}^5$ computations were used to prove the existence of invariant tori around some elliptic periodic orbits for hamiltonian and reversible systems. The approach is based on the classical KAM theorem for twist maps on the plane. We believe that the proposed algorithm has a wide spectrum of other applications.

2. Faá di Bruno formula

To effectively deal with the formulas involving the partial derivatives of the composition of maps we will extensively use a notation of multiindices, multipointers and submultipointers throughout the paper. In particular, when used, the variational equations can be written in a compact form.

2.1. Multiindices

By $\mathbb{N}$ we will denote the set of nonnegative integers, i.e. $\mathbb{N} = \{0, 1, 2, \ldots\}$.

Definition 1. *An element $\tau \in \mathbb{N}^n$ will be called a* multiindex.

For a sequence $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$ and a vector $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ we set

1. $|\alpha| = \alpha_1 + \cdots + \alpha_n$,
2. $\alpha! = \alpha_1 \cdot \alpha_2 \cdots \alpha_n$,
3. $x^\alpha = (x_1^{\alpha_1}, \ldots, x_n^{\alpha_n})$.

By $e_i^n \in \mathbb{N}^n$ we will denote

$$e_i^n = (0, 0, \ldots, 0, \overbrace{1}^{i}, 0, \ldots, 0, 0).$$

We will drop the index n (the dimension) in the symbol e_i^n when it is obvious from the context.

Put $\mathbb{N}_p^n := \{a \in \mathbb{N}^n : |a| = p\}$. For $\delta = (\delta_1, \ldots, \delta_k) \in \mathbb{N}^{n_1} \times \cdots \times \mathbb{N}^{n_k}$ we set

$$|\delta| = \sum_{i=1}^{k} |\delta_i|.$$

Let $f = (f_1, \ldots, f_m) : \mathbb{R}^n \to \mathbb{R}^m$ be sufficiently smooth. For $\alpha \in \mathbb{N}^n$ we set:

1. $D^\alpha f_i = \dfrac{\partial^{|\alpha|} f_i}{\partial x_1^{\alpha_1} \cdots \partial x_n^{\alpha_n}}$,
2. $D^\alpha f = (D^\alpha f_1, D^\alpha f_2, \ldots, D^\alpha f_m)$.

For a function $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ by $D^\alpha f_i(t, x)$ we will denote $D^\alpha f_i(t, \cdot)(x)$ and similarly

$$D^\alpha f(t, x) = (D^\alpha f_1(t, x), \ldots, D^\alpha f_n(t, x)).$$

This convention means that D^α always acts on x-variables.

2.2. Multipointers

For a fixed $n > 0$ and $p > 0$ we define:

$$\begin{aligned} \mathcal{N}_p^n &= \{(a_1, a_2, \ldots, a_p) \in \mathbb{N}^p : 1 \le a_1 \le \cdots \le a_p \le n\}, \\ \mathcal{N} = \mathcal{N}^n &= \bigcup_{p=1}^{\infty} \mathcal{N}_p^n. \end{aligned}$$

Definition 2. *An element of $\mathcal{N}^n$ will be called a* multipointer.

Remark 2. *A function*

$$\Lambda : \mathcal{N}_p^n \ni (a_1, \ldots, a_p) \to \sum_{i=1}^{p} e_{a_i}^n \in \mathbb{N}_p^n \tag{6}$$

is a bijection.

Let $f = (f_1, \ldots, f_m) : \mathbb{R}^n \to \mathbb{R}^m$ be sufficiently smooth. For $a \in \mathcal{N}_p^n$ we set:

1. $D_a f_i = \dfrac{\partial^p f_i}{\partial x_{a_1} \ldots \partial x_{a_p}}$,

2. $D_a f = (D_a f_1, \ldots, D_a f_m)$.

For a function $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ by $D_a f_i(t,x)$ we will denote $D_a f_i(t,\cdot)(x)$. In the light of the above notations $D_\alpha f = D^{\Lambda(\alpha)} f$.

For $a = (a_1, a_2, \ldots, a_n) \in \mathbb{N}_p^n$ and $b = (b_1, b_2, \ldots, b_n) \in \mathbb{N}_q^n$ we define

$$a + b = (a_1 + b_1, \ldots, a_n + b_n) \in \mathbb{N}_{p+q}^n.$$

For $\alpha \in \mathcal{N}_p^n$ and $\beta \in \mathcal{N}_q^n$ we define

$$\alpha + \beta = \Lambda^{-1}\left(\Lambda(\alpha) + \Lambda(\beta)\right) \in \mathcal{N}_{p+q}^n.$$

By $\leq$ we will denote a linear order (the lexicographical order) in $\mathcal{N}$ defined in the following way. For $a \in \mathcal{N}_p^n$ and $b \in \mathcal{N}_q^n$

$$(a \leq b) \Longleftrightarrow \begin{cases} \text{either } \exists i, i \leq p, i \leq q, a_i < b_i \text{ and } a_j = b_j \text{ for } j < i \\ \text{or } p \leq q \text{ and } a_i = b_i \text{ for } i = 1, \ldots, p. \end{cases} \tag{7}$$

Definition 3. *For $k \leq p$ we set*

$$\mathcal{N}^p(k) := \{(\delta_1, \ldots, \delta_k) \in (\mathcal{N}^p)^k : \delta_1 \leq \cdots \leq \delta_k, \delta_1 + \cdots + \delta_k = (1, 2, \ldots, p)\}. \tag{8}$$

We will use $\mathcal{N}^p(k)$ extensively in the next section. It will be used to label terms in $D^\alpha f_i(\varphi(t,x))$. Observe that for $p > 0$

$$\mathcal{N}^p(1) = \{(1, 2, \ldots, p)\},$$
$$\mathcal{N}^p(p) = \{((1), (2), \ldots, (p))\}.$$

One can construct all elements of $\mathcal{N}^p(k)$ using the following recursive procedure. From the definition of $\mathcal{N}^p(k)$ it follows that if $(\delta_1, \ldots, \delta_{m-1}) \in \mathcal{N}^{p-1}(m-1)$, then $(\delta_1, \ldots, \delta_{m-1}, (p)) \in \mathcal{N}^p(m)$ (notice that the order is preserved). Similarly, if $(\delta_1, \ldots, \delta_m) \in \mathcal{N}^{p-1}(m)$, then

$$(\delta_1, \ldots, \delta_{s-1}, \delta_s + (p), \delta_{s+1}, \ldots, \delta_m) \in \mathcal{N}^p(m)$$

and again the order of elements is preserved. Hence, for $p > 2$ and $1 < k < p$ we have $\mathcal{N}^p(k) = A \cup B$, where:

$$\begin{aligned} A &= \left\{(\delta_1, \ldots, \delta_{k-1}, (p)) : (\delta_1, \ldots, \delta_{k-1}) \in \mathcal{N}^{p-1}(k-1)\right\}, \\ B &= \bigcup_{s=1}^{k} \left\{(\delta_1, \ldots, \delta_{s-1}, \delta_s + (p), \delta_{s+1}, \ldots, \delta_k) : (\delta_1, \ldots, \delta_k) \in \mathcal{N}^{p-1}(k)\right\} \end{aligned} \tag{9}$$

and the sets A and B are disjoint.

Another way to generate all elements of $\mathcal{N}^p(k)$ can be described as follows:

- decompose the set $\{1, 2, \ldots, p\}$ into k nonempty and disjoints sets Δ_i, $i = 1, \ldots, k$,
- sort each Δ_i and permute Δ_i's to obtain $\min(\Delta_1) < \min(\Delta_2) < \cdots < \min(\Delta_k)$,

- define δ_i to be an ordered set consisting of all elements of Δ_i for $i = 1, \ldots, k$.

DEFINITION 4. *For an arbitrary $a \in \mathcal{N}_p^n$ and $\delta \in \mathcal{N}_k^p$ such that $k \leq p$ we define a* submultipointer *$a_\delta \in \mathcal{N}_k^n$ by $(a_\delta)_i = a_{\delta_i}$ for $i = 1, \ldots, k$, which can be expressed using Λ as follows:*

$$a_\delta := \Lambda^{-1}\left(\sum_{i=1}^{k} e^n_{a_{\delta_i}}\right) \in \mathcal{N}_k^n.$$

2.3. The variational equations

Consider an ODE $x' = f(x)$, where f is $\mathcal{C}^{K+1}$. Let $\varphi : D \subset \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ be a local dynamical system induced by $x' = f(x)$. It is a well-known fact that $\varphi \in \mathcal{C}^K$ and one can derive the equations for partial derivatives of φ by differentiating equation $\frac{\partial \varphi}{\partial t}(t, x) = f(\varphi(t, x))$ with respect to the initial condition x. As a result we obtain a system of so-called equations for variations, the size of which depends on the order r of the partial derivatives we intend to compute. An example of such a system for $r = 2$ is given by (2–4) with the initial conditions given by (5).

The equations for the higher order partial derivatives written in a compact form using multipointers and multiindices are given by the Faá di Bruno formula.

LEMMA 1 ([8], Faá di Bruno formula). *For any p-times continuously differentiable functions $f, g : \mathbb{R}^n \to \mathbb{R}^n$ and $a \in \mathcal{N}_p^n$ we have*

$$D_a(f \circ g) = \sum_{k=1}^{p} \sum_{i_1,\ldots,i_k=1}^{n} \left(D^{e_{i_1}+\cdots+e_{i_k}} f_i\right) \circ g \sum_{(\delta_1,\ldots,\delta_k)\in\mathcal{N}^p(k)} \prod_{j=1}^{k} D_{a_{\delta_j}} g_{i_j}. \qquad (10)$$

Proof: In the proof the functions $D^{e_{i_1}+\cdots+e_{i_k}} f_i$ are always evaluated at $g(x)$, and various partial derivatives of g are always evaluated at x, therefore, to simplify formulae, the arguments will always be dropped.

Put $F = f \circ g$. We prove the lemma by induction on $p = |a|$. If $p = 1$, then $a = (c)$ for some $c \in \{1, \ldots, n\}$ and (15) becomes

$$D_{(c)}F = \sum_{s=1}^{n} \frac{\partial f_i}{\partial x_s} \frac{\partial g_s}{\partial x_c} = \sum_{s=1}^{n} D^{e_s} f_i \cdot D_{(c)} g_s.$$

Assume (15) holds true for $p-1$, $p > 1$. Let us fix $a \in \mathcal{N}_p^n$. We have $a = b + (c)$, where $b = (a_1, \ldots, a_{p-1}) \in \mathcal{N}_{p-1}^n$ and $c = a_p$. Since (15) is satisfied for $p - 1$, we

have

$$
\begin{aligned}
D_a F_i &= D_{(c)}\left(D_b F_i\right) \\
&= D_{(c)}\left(\sum_{k=1}^{p-1} \sum_{\substack{i_1,\ldots,i_k=1 \\ \beta:=e_{i_1}+\cdots+e_{i_k}}}^{n} D^{\beta} f_i \sum_{(\delta_1,\ldots,\delta_k)\in\mathcal{N}^{p-1}(k)} \prod_{j=1}^{k} D_{b_{\delta_j}} g_{i_j}\right) \\
&= \sum_{k=1}^{p-1} \sum_{\substack{i_1,\ldots,i_{k+1}=1 \\ \beta:=e_{i_1}+\cdots+e_{i_{k+1}}}}^{n} D^{\beta} f_i \cdot D_{(c)} g_{i_{k+1}} \sum_{(\delta_1,\ldots,\delta_k)\in\mathcal{N}^{p-1}(k)} \prod_{j=1}^{k} D_{b_{\delta_j}} g_{i_j} \\
&+ \sum_{k=1}^{p-1} \sum_{\substack{i_1,\ldots,i_k=1 \\ \beta:=e_{i_1}+\cdots+e_{i_k}}}^{n} D^{\beta} f_i \sum_{(\delta_1,\ldots,\delta_k)\in\mathcal{N}^{p-1}(k)} \sum_{s=1}^{k} D_{b_{\delta_s}+(c)} g_{i_s} \prod_{\substack{j=1, \\ j\neq s}}^{k} D_{b_{\delta_j}} g_{i_j}.
\end{aligned}
$$

For $k = 1, \ldots, p$ we set

$$
T_k := \sum_{i_1,\ldots,i_k=1}^{n} D^{e_{i_1}+\cdots+e_{i_k}} f_i \sum_{(\delta_1,\ldots,\delta_k)\in\mathcal{N}^{p}(k)} \prod_{j=1}^{k} D_{a_{\delta_j}} g_{i_j}. \tag{11}
$$

Now our goal is to prove that:

$$
D_a F_i = \sum_{k=1}^{p} T_k. \tag{12}
$$

Our strategy of proof is as follows. We will define $S_1, \ldots, S_p$, such that

$$
D_a F_i = \sum_{k=1}^{p} S_k \tag{13}
$$

and we will show that $S_i = T_i$ for $i = 1, \ldots, p$.

We set

$$
\begin{aligned}
S_1 &= \sum_{k=1}^{1} \sum_{\substack{i_1,\ldots,i_k=1 \\ \beta:=e_{i_1}+\cdots+e_{i_k}}}^{n} D^{\beta} f_i \sum_{(\delta_1,\ldots,\delta_k)\in\mathcal{N}^{p-1}(k)} \sum_{s=1}^{k} D_{b_{\delta_s}+(c)} g_{i_s} \prod_{\substack{j=1, \\ j\neq s}}^{k} D_{b_{\delta_j}} g_{i_j}, \\
S_p &= \sum_{k=p-1}^{p-1} \sum_{\substack{i_1,\ldots,i_{k+1}=1 \\ \beta:=e_{i_1}+\cdots+e_{i_{k+1}}}}^{n} D^{\beta} f_i \cdot D_{(c)} g_{i_{k+1}} \sum_{(\delta_1,\ldots,\delta_k)\in\mathcal{N}^{p-1}(k)} \prod_{j=1}^{k} D_{b_{\delta_j}} g_{i_j}.
\end{aligned}
$$

For $m = 2, 3, \ldots, p-1$ we set

$$
\begin{aligned}
S_m &= \sum_{k=m-1}^{m-1} \sum_{\substack{i_1,\ldots,i_{k+1}=1 \\ \beta:=e_{i_1}+\cdots+e_{i_{k+1}}}}^{n} D^{\beta} f_i \cdot D_{(c)} g_{i_{k+1}} \sum_{(\delta_1,\ldots,\delta_k)\in\mathcal{N}^{p-1}(k)} \prod_{j=1}^{k} D_{b_{\delta_j}} g_{i_j} \\
&+ \sum_{k=m}^{m} \sum_{\substack{i_1,\ldots,i_k=1 \\ \beta:=e_{i_1}+\cdots+e_{i_k}}}^{n} D^{\beta} f_i \sum_{(\delta_1,\ldots,\delta_k)\in\mathcal{N}^{p-1}(k)} \sum_{s=1}^{k} D_{b_{\delta_s}+(c)} g_{i_s} \prod_{\substack{j=1, \\ j\neq s}}^{k} D_{b_{\delta_j}} g_{i_j}.
\end{aligned}
$$

It remains to show that $S_i = T_i$ for $i = 1, \dots, p$. Consider first $i = 1$. Recall that $\mathcal{N}^{p-1}(1) = \{(1, 2, \dots, p-1)\}$, hence

$$S_1 = \sum_{s=1}^{n} D^{e_s} f_i \cdot D_{b+(c)} g_s = \sum_{s=1}^{n} D^{e_s} f_i \cdot D_a g_s.$$

Therefore

$$S_1 = T_1. \tag{14}$$

Consider now $i = p$. For an arbitrary $s > 0$ $\mathcal{N}^s(s)$ contains only one element $((1), (2), \dots, (s))$. Therefore we obtain

$$\begin{aligned} S_p &= \sum_{i_1,\dots,i_p=1}^{n} D^{e_{i_1}+\dots+e_{i_p}} f_i \cdot D_{(c)} g_{i_p} \sum_{(\delta_1,\dots,\delta_{p-1})\in\mathcal{N}^{p-1}(p-1)} \prod_{j=1}^{p-1} D_{b_{\delta_j}} g_{i_j} \\ &= \sum_{i_1,\dots,i_p=1}^{n} D^{e_{i_1}+\dots+e_{i_p}} f_i \cdot D_{(c)} g_{i_p} \prod_{j=1}^{p-1} D_{b_j} g_{i_j}. \end{aligned}$$

Since $a = b + (c)$, where $c = (a_p)$, we have

$$\begin{aligned} S_p &= \sum_{i_1,\dots,i_p=1}^{n} D^{e_{i_1}+\dots+e_{i_p}} f_i \prod_{j=1}^{p} D_{a_j} g_{i_j} \\ &= \sum_{i_1,\dots,i_p=1}^{n} D^{e_{i_1}+\dots+e_{i_p}} f_i \sum_{(\delta_1,\dots,\delta_p)\in\mathcal{N}^p(p)} \prod_{j=1}^{p} D_{a_{\delta_j}} g_{i_j} = T_p. \end{aligned}$$

Consider now $m = 2, 3, \dots, p-1$. We have

$$\begin{aligned} S_m = &\sum_{i_1,\dots,i_m=1}^{n} D^{e_{i_1}+\dots+e_{i_m}} f_i \cdot D_{(c)} g_{i_m} \sum_{(\delta_1,\dots,\delta_{m-1})\in\mathcal{N}^{p-1}(m-1)} \prod_{j=1}^{m-1} D_{b_{\delta_j}} g_{i_j} \\ &+ \sum_{i_1,\dots,i_m=1}^{n} D^{e_{i_1}+\dots+e_{i_m}} f_i \sum_{(\delta_1,\dots,\delta_m)\in\mathcal{N}^{p-1}(m)} \sum_{s=1}^{m} D_{b_{\delta_s}+(c)} g_{i_s} \prod_{\substack{j=1,\\ j\neq s}}^{m} D_{b_{\delta_j}} g_{i_j}. \end{aligned}$$

Using the decomposition $\mathcal{N}^p(m) = A \cup B$ as in (9) we obtain

$$\begin{aligned} S_m &= \sum_{i_1,\dots,i_m=1}^{n} D^{e_{i_1}+\dots+e_{i_m}} f_i \sum_{(\delta_1,\dots,\delta_{m-1},\delta_m=(p))\in A} \prod_{j=1}^{m} D_{a_{\delta_j}} g_{i_j} \\ &+ \sum_{i_1,\dots,i_m=1}^{n} D^{e_{i_1}+\dots+e_{i_m}} f_i \sum_{(\delta_1,\dots,\delta_m)\in B} \prod_{j=1}^{m} D_{a_{\delta_j}} g_{i_j} \\ &= \sum_{i_1,\dots,i_m=1}^{n} D^{e_{i_1}+\dots+e_{i_m}} f_i \sum_{(\delta_1,\dots,\delta_m)\in\mathcal{N}^p(m)} \prod_{j=1}^{m} D_{a_{\delta_j}} g_{i_j} = T_m. \end{aligned}$$

We have shown that $T_i = S_i$ for $i = 1, \dots, p$. This finishes the proof. ∎

From the above lemma we have immediately

LEMMA 2. *Assume $f \in \mathcal{C}^{r+1}$ and let $\varphi : \mathbb{R} \times \mathbb{R}^n \multimap \mathbb{R}^n$ be a local dynamical system induced by $x' = f(x)$. Then for $a \in \mathcal{N}_p^n$ such that $p \leq r$ holds*

$$\frac{d}{dt} D_a \varphi_i = \sum_{k=1}^{p} \sum_{i_1,\ldots,i_k=1}^{n} \left(D^{e_{i_1}+\cdots+e_{i_k}} f_i\right) \circ \varphi \sum_{(\delta_1,\ldots,\delta_k)\in\mathcal{N}^p(k)} \prod_{j=1}^{k} D_{a_{\delta_j}} \varphi_{i_j} \tag{15}$$

for $i = 1, \ldots, n$.

Formula (15) could be seen as a direct application of the chain rule for composition of multivariate power series. Using the automatic differentiation tools [7, 5, 20] one can efficiently nonrigorously integrate ODEs together with higher order variational equations by means of floating point arithmetic.

The main goal of this paper is to present an efficient rigorous solver for higher order variational equations which takes into account the structure of the equations and the wrapping effect.

3. $\mathcal{C}^r$-Lohner algorithm

3.1. Why one needs an $\mathcal{C}^r$-algorithm?

In the literature there exist several effective algorithms for the computation of the rigorous bounds for the solutions of ordinary differential equations, including the Lohner method [13], the Hermite–Obreschkoff algorithm [18] or the Taylor model [3]. For the $\mathcal{C}^r$-computations the number of equations to solve is equal to $n\binom{n+r}{n}$, hence, even for $r = 1$ the direct application of such algorithms to the equations for variations (16) leads to the integration in the high dimensional space and is usually inefficient. Let us recall after [31, Sec. 6] the basic reason for this. In order to have a good control over the expansion rate of the set of the initial conditions during a time step these algorithms, while being $\mathcal{C}^0$, are $\mathcal{C}^1$ 'internally' (or higher for the Taylor models), because they solve non-rigorously the equations for $(\frac{\partial\varphi}{\partial x})$ – the variational matrix of the flow. This effectively squares the dimension of the phase space of the equation and impacts heavily on the computation time. But as it was observed in [31] the equations for the partial derivatives of the flow can be seen as the non-autonomous and nonhomogenous linear system of equations, therefore we do not need any additional equations for variations for them. As a result, the dimension of the effective phase space for our $\mathcal{C}^r$-algorithm is given by $n\binom{n+r}{n}$ instead of the square of this number.

Another important aspect of the proposed algorithm is the fact that the Lohner-type control of the wrapping effect is done separately for x-variables and variables $D_a\varphi$. This feature is not present in the naïve application of $\mathcal{C}^0$ algorithm to the

system of variational equations and it turns out that it often practically switches off the control of the wrapping effect on x-variables, as various choices used in this control become dominated by the $D_a\varphi$-variables.

In Section 7 we will give a detailed comparison of the $\mathcal{C}^0$-solver applied to the equations of variations and our $\mathcal{C}^r$-solver.

3.2. An outline of the algorithm

Let us fix $r \leq K$ and consider the following system of differential equations

$$\begin{cases} \dfrac{d}{dt}\varphi = f \circ \varphi \\ \dfrac{d}{dt} D_a\varphi = \displaystyle\sum_{k=1}^{d} \sum_{i_1,\ldots,i_k=1}^{n} \left(D^{e_{i_1}+\cdots+e_{i_k}} f\right) \circ \varphi \sum_{(\delta_1,\ldots,\delta_k)\in\mathcal{N}^d(k)} \prod_{j=1}^{k} D_{a_{\delta_j}} \varphi_{i_j} \end{cases} \tag{16}$$

for all $a \in \mathcal{N}_d^n$, $d = 1, \ldots, r$.

The initial conditions for (16) are

$$\begin{cases} \varphi(0, x_0) & \in [x_0] \subset \mathbb{R}^n, \\ D\varphi(0, x_0) & = \mathrm{Id}, \\ D_a\varphi(0, x_0) & = 0, \qquad \text{for } a \in \mathcal{N}_2^n \cup \ldots \cup \mathcal{N}_r^n. \end{cases} \tag{17}$$

In the sequel we will use the following notations:

- if a solution of the system (16) is defined for $t > 0$ and some $x_0 \in \mathbb{R}^n$, then for $a \in \mathcal{N}$ by $V_a(t, x_0)$ we denote $D_a\varphi(t, x_0)$,
- for $[x_0] \subset \mathbb{R}^n$ by $[V_a(t, [x_0])]$ we will denote a set for which we have $V_a(t, [x_0]) \subset [V_a(t, [x_0])]$. This set is obtained using a rigorous numerical routine described below.

The $\mathcal{C}^r$-Lohner algorithm is a modification of the $\mathcal{C}^1$-Lohner algorithm [31]. One step of the $\mathcal{C}^r$-Lohner is a shift along the trajectory of the system (16) with the following input and output data.

Input data:

- t_k - current time,
- h_k - time step,
- $[x_k] \subset \mathbb{R}^n$, such that $\varphi(t_k, [x_0]) \subset [x_k]$,
- $[V_{k,a}] = [V_{k,a}(t_k, [x_0])] \subset \mathbb{R}^n$, such that $D_a\varphi(t_k, [x_0]) \subset [V_{k,a}]$ for $a \in \mathcal{N}_1^n \cup \ldots \cup \mathcal{N}_r^n$.

Output data:

- $t_{k+1} = t_k + h_k$ - new current time,
- $[x_{k+1}] \subset \mathbb{R}^n$, such that $\varphi(t_{k+1}, [x_0]) \subset [x_{k+1}]$,
- $[V_{k+1,a}] = [V_{k+1,a}(t_{k+1}, [x_0])] \subset \mathbb{R}^n$, such that $D_a\varphi(t_{k+1}, [x_0]) \subset [V_{k+1,a}]$ for $a \in \mathcal{N}_1^n \cup \ldots \cup \mathcal{N}_r^n$.

We will often skip the arguments of $V_{k,a}$ when they are obvious from the context.

The values of $[x_{k+1}]$ and $[V_{k+1,a}]$, $a \in \mathcal{N}_1^n$ are computed using one step of the $\mathcal{C}^1$-Lohner algorithm. After this is done, we perform the following operations to compute $[V_{k+1,a}]$ for $a \in \mathcal{N}_2^n \cup \ldots \cup \mathcal{N}_r^n$

1. Find a rough enclosure for $D_a\varphi([0, h_k], [x_k])$.

2. Compute $[V_{k+1,a}]$. This will also involve some rearrangement computations to reduce the wrapping effect for V [15, 13].

4. Computation of a rough enclosure for $D_a\varphi$

For a fixed multipointer $a \in \mathcal{N}_d^n$ Eq. (16) can be written as follows:

$$\frac{d}{dt} D_a\varphi(t, x) = B_a(t, x) + A(t, x) D_a\varphi(t, x), \tag{18}$$

where

$$\begin{aligned} B_a &= \sum_{k=2}^{d} \sum_{i_1,\ldots,i_k=1}^{n} \left(D^{e_{i_1}+\cdots+e_{i_k}} f\right) \circ \varphi \sum_{(\delta_1,\ldots,\delta_k)\in\mathcal{N}^d(k)} \prod_{j=1}^{k} D_{a_{\delta_j}} \varphi_{i_j}, \\ A &= Df \circ \varphi. \end{aligned} \tag{19}$$

The procedure for computing the rough enclosure is based on the notion of the logarithmic norm.

Definition 5. *[9] For a square matrix A the logarithmic norm $\mu(A)$ is defined as a limit*

$$\mu(A) = \limsup_{h\to 0^+} \frac{\|\mathrm{Id} + Ah\| - 1}{h},$$

where $\|\cdot\|$ is a given matrix norm.

The formulae for the logarithmic norm of a real matrix in the most frequently used norms are (see [9]):

1. for $\|x\|_1 = \sum_i |x_i|$, $\mu(A) = \max_j (a_{jj} + \sum_{i\neq j} |a_{ij}|)$,
2. for $\|x\|_2 = \sqrt{\sum_i |x_i|^2}$, $\mu(A)$ is equal to the largest eigenvalue of $(A + A^T)/2$,

3. for $\|x\|_\infty = \max_i |x_i|$, $\mu(A) = \max_i(a_{ii} + \sum_{j\neq i} |a_{ij}|)$.

In order to find bounds for $D_a\varphi$ we use the following theorem [9, Thm. I.10.6]

THEOREM 3. *Let $x(t)$ be a solution of a differential equation*

$$x'(t) = f(t, x(t)), \quad x \in \mathbb{R}^n. \tag{20}$$

Let $\nu(t)$ be a piecewise differentiable function with values in $\mathbb{R}^n$. Assume that

$$\mu\left(\frac{\partial f}{\partial x}(t,\eta)\right) \leq l(t) \quad \text{for } \eta \in [x(t), \nu(t)],$$
$$|\nu'(t) - f(t, \nu(t))| \leq \delta(t).$$

Then for $t \geq t_0$ we have

$$|x(t) - \nu(t)| \leq e^{L(t)} \left(|x(t_0) - \nu(t_0)| + \int_{t_0}^{t} e^{-L(s)}\delta(s)ds \right), \tag{21}$$

with $L(t) = \int_{t_0}^{t} l(\tau)d\tau$.

We apply the above theorem to Equation (18) to obtain

LEMMA 3. *Let us fix $x \in \mathbb{R}^n$. Assume that $|B_a(t,x)| \leq \delta(t)$ and $\mu(A(t,x)) \leq l(t)$, then for $t > t_0$ holds*

$$|D_a\varphi(t,x)| \leq |D_a\varphi(t_0,x)|e^{L(t)} + e^{L(t)} \int_{t_0}^{t} e^{-L(\tau)}\delta(\tau)d\tau \tag{22}$$

with $L(t) = \int_{t_0}^{t} l(\tau)d\tau$.

Proof: Consider Eq. (18) and a homogenous problem for (18)

$$\frac{d}{dt}w = f(t,w) := A(t,x)\cdot w, \qquad w \in \mathbb{R}^n. \tag{23}$$

Using Theorem 3 we can estimate the difference between any solution of (23), w, and a solution of (18), denoted by $D_a\varphi$.

$$|D_a\varphi(t) - w(t)| \leq |D_a\varphi(t_0) - w(t_0)|e^{L(t)} + e^{L(t)} \int_{t_0}^{t} e^{-L(\tau)}\delta(\tau)d\tau. \tag{24}$$

After a substitution $w(t) = 0$, which is a solution of the homogenous equation, we obtain our assertion. ∎

Usually we do not have any control over the time dependence of δ and l, hence we will use the following

LEMMA 4. *Assume that $|B_a(t,x)| \leq \delta$ and $\mu(A(t,x)) \leq l$ for $t \in [0,h]$ then for $t \in [0,h]$ we have*

$$|D_a\varphi(t,x)| \leq |D_a\varphi(0,x)| \max(1, e^{hl}) + \delta\frac{e^{lt}-1}{l}, \quad \text{if } l \neq 0, \tag{25}$$

or

$$|D_a\varphi(t,x)| \leq |D_a\varphi(0,x)| + \delta t, \quad \text{when } l = 0. \tag{26}$$

4.1. The procedure for the computation of the rough enclosure for V

For $a \in \mathcal{N}_1^n \cup \ldots \cup \mathcal{N}_r^n$ by $[E_a]$ we will denote the rough enclosure for the corresponding variational equation. The procedure for the computation of the rough enclosure $[E_a]$ is iterative, which means that given the rough enclosure for $\varphi([0,h_k],[x_k])$ and the rough enclosures $D_a\varphi([0,h_k],[x_k])$ for all $a \in \mathcal{N}_1^n \cup \ldots \cup \mathcal{N}_p^n$ we are able to compute the rough enclosure for $D_a\varphi([0,h_k],[x_k])$ for $a \in \mathcal{N}_{p+1}^n$.

The procedures for the computation of the rough enclosures of $\varphi([0,h_k],[x_k])$ and $D_a\varphi([0,h_k],[x_k])$ for $a \in \mathcal{N}_1^n$ were given in [31]. Below we present an algorithm for computing $[E_a]$ for $a \in \mathcal{N}_2^n \cup \ldots \cup \mathcal{N}_r^n$.

Input parameters:

- h_k – time step,
- $[x_k] \subset \mathbb{R}^n$ – the current value of $x = \varphi(t_k,[x_0])$,
- $[E_0] \subset \mathbb{R}^n$ – a compact and convex set such that $\varphi([0,h_k],[x_k]) \subset [E_0]$,
- $[E_a] \subset \mathbb{R}^n$, $a \in \mathcal{N}_1^n \cup \ldots \cup \mathcal{N}_p^n$ such that $D_a\varphi([0,h_k],[x_k]) \subset [E_a]$ for $a \in \mathcal{N}_1^n \cup \ldots \cup \mathcal{N}_p^n$.

Output:

- $[E_a] \subset \mathbb{R}^n$, $a \in \mathcal{N}_{p+1}^n$ such that

$$D_a\varphi([0,h_k],[x_k]) \subset [E_a].$$

Before we present an algorithm let us observe that for a fixed $a \in \mathcal{N}_{p+1}^n$, B_a defined in (19) could be seen as a multivariate function of t, x and $V_b = D_b\varphi$ for $b \in \mathcal{N}_1^n \cup \ldots \cup \mathcal{N}_p^n$. More precisely, put $m_p := \sharp\left\{\mathcal{N}_1^n \cup \ldots \cup \mathcal{N}_p^n\right\}$, where $\sharp$ stands for the number of elements of a set. Recall that we have defined by (7) a linear order in $\mathcal{N}^n$. Hence, there is a unique sequence of multipointers $b_1,\ldots,b_{m_p}$, such that $b_i \in \mathcal{N}_1^n \cup \ldots \cup \mathcal{N}_p^n$ for $i = 1,\ldots,m_p$, $b_1 \le b_2 \le \cdots \le b_{m_p}$ and $b_i \neq b_j$ for $i \neq j$.

Let us define

$$\tilde{B}_a : \mathbb{R} \times (\mathbb{R}^n)^{m_p+1} \to \mathbb{R}^n,$$
$$F_a : \mathbb{R} \times (\mathbb{R}^n)^{m_p+1} \to \mathbb{R}^n$$

by

$$\tilde{B}_a(t,x,v_{b_1},\ldots,v_{b_{m_p}}) = \sum_{k=2}^{p+1} \sum_{i_1,\ldots,i_k=1}^{n} D^{e_{i_1}+\cdots+e_{i_k}} f(\varphi(t,x)) \sum_{(\delta_1,\ldots,\delta_k)\in\mathcal{N}^{p+1}(k)} \prod_{j=1}^{k} \left(v_{a_{\delta_j}}\right)_{i_j} \quad (27)$$

and

$$F_a(t,x,v_{b_1},\ldots,v_{b_m}) = \tilde{B}_a(t,x,v_{b_1},\ldots,v_{b_m}) + Df(\varphi(t,x))V_a(t,x). \quad (28)$$

Algorithm:
To compute $[E_a]$ for $a \in \mathcal{N}_{p+1}^n$ we proceed as follows:

1. Find $l \geq \left(\max_{x\in[E_0]} \mu\left(Df(x)\right)\right)$.

2. Compute $\delta_a \geq \max \|\tilde{B}_a\|$, i.e.

$$\delta_a \geq \max_{(x,v_{b_1},\ldots,v_{b_{m_p}})\in[E_0]\times[E_{b_1}]\times\cdots\times[E_{b_{m_p}}]} \left\|\tilde{B}_a(0,x,v_{b_1},\ldots,v_{b_{m_p}})\right\|.$$

For example, if $a=(j,c)\in\mathcal{N}_2^n$, then δ_a should be such that

$$\delta_a \geq \max_{x\in[E_0],v_1\in[E_{(1)}],\ldots,v_n\in[E_{(n)}]} \left\|\sum_{r,s=1}^{n} \frac{\partial^2 f}{\partial x_r \partial x_s}(x)\left(v_j\right)_s\left(v_c\right)_r\right\|.$$

3. Define $[E_a]_i = [-1,1]\delta_a \frac{e^{lt}-1}{l}$, for $i=1,\ldots,n$, where $[E_a]_i$ denotes i-th coordinate of $[E_a]$.

One can refine the obtained enclosure by

$$[E_a] \quad := \quad \left([0,h_k]F_a\left(0,[E_0],[E_{b_1}],\ldots,[E_{b_{m_p}}]\right)\right)\cap[E_a].$$

Indeed, from (17) we have $D_a\varphi_i(0,x_0)=0$ for $i=1,\ldots,n$, $x_0\in[E_0]$ and for $t\in[0,h_k]$ we have

$$\begin{aligned} D_a\varphi_i(t,x_0) &= D_a\varphi_i(t,x_0) - D_a\varphi_i(0,x_0) \\ &= t\left(F_a\right)_i\left(\theta_i,x_0,D_{b_1}\varphi(\theta_i,x_0),\ldots,D_{b_{m_p}}\varphi(\theta_i,x_0)\right) \\ &= t\left(F_a\right)_i\left(0,\varphi(\theta_i,x_0),D_{b_1}\varphi(\theta_i,x_0),\ldots,D_{b_{m_p}}\varphi(\theta_i,x_0)\right) \end{aligned}$$

for some $\theta_i\in[0,t]\subset[0,h_k]$. In the above we have used the fact that

$$F_a(t,x,v_1,\ldots,v_{m_p}) = F_a(0,\varphi(t,x),v_1,\ldots,v_{m_p}).$$

Since $\varphi(\theta_i,x_0)\in[E_0]$ and $D_{b_j}\varphi(\theta_i,x_0)\in[E_{b_j}]$ for $j=1,\ldots,m_p$ we get

$$D_a\varphi_i(t,x_0)\in[0,h_k]\left(F_a\right)_i\left(0,[E_0],[E_{b_1}],\ldots,[E_{b_{m_p}}]\right).$$

5. Computation of $[V_{k+1}]$

5.1. Composition formulae

We apply the Faá di Bruno formula (10) to $f = \varphi(h_k, \cdot)$ and $g = \varphi(t_k, \cdot)$ to obtain

$$V_a(t_k + h_k, x_0) = \sum_{k=1}^{p} \sum_{i_1,\ldots,i_k=1}^{n} V_{\Lambda^{-1}(e_{i_1}+\ldots+e_{i_k})}(h_k, x_k) \sum_{(\delta_1,\ldots,\delta_k)\in\mathcal{N}^p(k)} \prod_{j=1}^{k} \left(V_{a_{\delta_j}}\right)_{i_j} (t_k, x_0)$$

for all $x_0 \in [x_0]$. Using notations $[V_{k+1,a}] := [V_a(t_k + h_k, [x_0])]$ and $[V_{k,a}] = [V_a(t_k, [x_0])]$ we can rewrite the above equation as

$$[V_{k+1,a}] = \sum_{k=1}^{p} \sum_{i_1,\ldots,i_k=1}^{n} V_{\Lambda^{-1}(e_{i_1}+\ldots+e_{i_k})}(h_k, [x_k]) \sum_{(\delta_1,\ldots,\delta_k)\in\mathcal{N}^p(k)} \prod_{j=1}^{k} \left[V_{k,a_{\delta_j}}\right]_{i_j}, \tag{29}$$

where Λ is defined by (6).

5.2. The procedure for the computation of $[V_{k+1}]$

We introduce new parameters o_d – the order of the Taylor method used in computations of V_a for $a \in \mathcal{N}_d^n$. It makes sense to take $o_1 \geq o_2 \geq \cdots \geq o_r$.
Input parameters:

- h_k – time step,
- $[x_k] \subset \mathbb{R}^n$ – the current value of $x = \varphi(t_k, [x_0])$,
- $[V_{k,a}] \subset \mathbb{R}^n$ – a current value of $V_{k,a}(t_k, [x_0])$, for $a \in \mathcal{N}_1^n \cup \ldots \cup \mathcal{N}_r^n$,
- $[E_0] \subset \mathbb{R}^n$ compact and convex, such that $\varphi([0, h_k], [x_k]) \subset [E_0]$ – a rough enclosure for $[x_k]$,
- $[E_a] \subset \mathbb{R}^n$, compact and convex, such that $D_a\varphi([0, h_k], [x_k]) \subset [E_a]$, for $a \in \mathcal{N}_1^n \cup \ldots \cup \mathcal{N}_r^n$.

Output: $[V_{k+1,a}] \subset \mathbb{R}^n$ such that

$$V_a(t_k + h_k, x_0) \in [V_{k+1,a}] \tag{30}$$

for $x_0 \in [x_0]$ and $a \in \mathcal{N}_1^n \cup \ldots \cup \mathcal{N}_r^n$.

Algorithm: We compute $[V_{k+1}]$ as follows:

1. Computation of $V_a(h_k,[x_k])$ using the Taylor method for Eq. (16), i.e., for $a\in\mathcal{N}_p^n$ we compute

$$\begin{aligned}[F_a] &= \sum_{i=1}^{o_p}\frac{h_k^i}{i!}\frac{d^{i-1}}{dt^{i-1}}F_a(0,[x_k],V_{b_1},\ldots,V_{b_{m_{p-1}}}) \\ &+ \frac{h^{o_p+1}}{(o_p+1)!}\frac{d^{o_p}}{dt^{o_p}}F_a(0,[E_0],[E_{b_1}],\ldots,[E_{b_{m_{p-1}}}]),\end{aligned} \tag{31}$$

where $V_{b_i}=0$ for $b_i\in\mathcal{N}_2^n\cup\ldots\cup\mathcal{N}_{p-1}^n$ and $V_{(j)}=e_j^n$ for $j=1,\ldots,n$. Observe that

$$V_a(h_k,[x_k])\subset[F_a]. \tag{32}$$

Indeed, using the Taylor series expansion we obtain that for $x_k\in[x_k]$ and $j=1,\ldots,n$ holds

$$\begin{aligned}(V_a)_j(h_k,x_k) = &\sum_{i=1}^{o_p}\frac{h_k^i}{i!}\frac{d^{i-1}}{dt^{i-1}}(F_a)_j(0,x_k,V_{b_1}(0,x_k),\ldots,V_{b_{m_{p-1}}}(0,x_k)) \\ &+\frac{h^{o_p+1}}{(o_p+1)!}\frac{d^{o_p}}{dt^{o_p}}(F_a)_j(\theta_i,x_k,V_{b_1}(\theta_i,x_k),\ldots,V_{b_{m_{p-1}}}(\theta_i,x_k))\end{aligned}$$

for some $\theta_i\in[0,h_k]$. Observe, that

$$\begin{aligned}&\frac{d^{o_p}}{dt^{o_p}}(F_a)_j(\theta_i,x_k,V_{b_1}(\theta_i,x_k),\ldots,V_{b_{m_{p-1}}}(\theta_i,x_k)) = \\ &\qquad\frac{d^{o_p}}{dt^{o_p}}(F_a)_j(0,\varphi(\theta_i,x_k),V_{b_1}(\theta_i,x_k),\ldots,0,V_{b_{m_{p-1}}}(\theta_i,x_k)).\end{aligned}$$

Since $\varphi(\theta_i,x_k)\in[E_0]$ and $V_{b_s}(\theta_i,x_k)\in[E_{b_s}]$ for $s=1,\ldots,m_{p-1}$ we obtain our assertion.

2. The composition. Put

$$[J_k]:=([F_{(1)}],\ldots,[F_{(n)}])^T.$$

Using (29) for $a\in\mathcal{N}_p^n$ we have

$$[V_{k+1,a}]=[\alpha_a]+[J_k]\cdot[V_{k,a}], \tag{33}$$

where

$$[\alpha_a]=\sum_{k=2}^{p}\sum_{i_1,\ldots,i_k=1}^{n}[F_{\Lambda^{-1}(e_{i_1}+\ldots+e_{i_k})}]\sum_{(\delta_1,\ldots,\delta_k)\in\mathcal{N}^p(k)}\prod_{j=1}^{k}\left[V_{k,a_{\delta_j}}\right]_{i_j}. \tag{34}$$

5.3. Rearrangement for V_a – the evaluation of Equation (33)

It is a well-known fact that a direct evaluation of Eq. (33) leads to the wrapping effect [15, 13]. To avoid it, following the work of Lohner [13] we will use the scheme proposed in [31] for the $\mathcal{C}^1$-algorithm.

Namely, observe that Eq. (33) has exactly the same structure as the propagation equations for $\mathcal{C}^1$-method (see [31, Section 3]). Moreover, all vectors $V_{k,a}$, for $a \in \mathcal{N}_1^n \cup \ldots \mathcal{N}_r^n$ 'propagate' by the same $[J_k]$ as does the variational part in [31], hence it makes sense to use the same approach.

To be more precise, each set $[V_{k,a}]$ (for $a \in \mathcal{N}_1^n \cup \ldots \cup \mathcal{N}_r^n$) is represented in the following form:

$$[V_{k,a}] = v_{k,a} + [B_k][r_{k,a}] + C_k[q_{k,a}],$$

where $[B_k]$ is an interval matrix, C_k is a point matrix, $v_{k,a}$ is a point vector and $r_{k,a}, q_{k,a}$ are interval vectors. Observe that $[B_k]$ and C_k are independent of a.

In the sequel we will drop the index a. Equation (33) leads to

$$[V_{k+1}] = [\alpha] + [J_k](v_k + [B_k][r_k] + C_k[q_k]). \tag{35}$$

Let $m([z])$ denotes a center of an interval object, i.e., $[z]$ is interval vector or interval matrix and let $\Delta([z]) = [z] - m([z])$.

Let $[Q]$ be an interval matrix which contains an orthogonal matrix. Usually, $[Q]$ is computed by the orthonormalisation of the columns of $m([J_k][B_k])$.

Let

$$\begin{aligned} [Z] &= [J_k]C_k, \\ C_{k+1} &= m([Z]), \\ [B_{k+1}] &= [Q]. \end{aligned}$$

Then we rearrange formula (35) as follows:

$$\begin{aligned} [s] &= [\alpha] + [J_k]v_k + \Delta([Z])[q_k], \\ v_{k+1} &= m([s]), \\ [q_{k+1}] &= [q_k], \\ [r_{k+1}] &= [Q^T]\Delta([s]) + \left([Q^T][J_k][B_k]\right)[r_k]. \end{aligned} \tag{36}$$

Summarizing, we can use the following data structure to represent $\varphi(t_k, [x_0])$ and $D_a\varphi(t_k, [x_0])$, for $a \in \mathcal{N}_1^n \cup \ldots \cup \mathcal{N}_r^n$:

type CnSet = **record**

v_0, r_0, q_0: IntervalVector;
C_0, B_0, C, B : IntervalMatrix;
$\{v_a, r_a, q_a :$ IntervalVector$\}_{a \in \mathcal{N}_1^n \cup \ldots \cup \mathcal{N}_r^n}$

end;

The set $\varphi(t_k, [x_0])$ is represented as $v_0 + B_0 r_0 + C_0 q_0$. The partial derivatives $D_a\varphi(t_k, [x_0])$ are represented as $v_a + Br_a + Cq_a$. The matrices B, C are common for all partial derivatives.

Notice, that if we start the $\mathcal{C}^r$ computation with an initial condition (17), then there is no Lipschitz part at the beginning for the partial derivatives. Hence, the initial values for C and B are set to the identity matrix and the initial values for q_a, r_a are set to zero.

If the interval vectors r_a become 'thick' (i.e. their diameters are larger than some threshold value) we can set a new Lipschitz part in our representation (it must be done simultaneously for all $D_a\varphi$) and reset r_a in the following way:

$$\begin{aligned} q_a &= r_a + (B^T C)q_a, \quad \text{for } a \in \mathcal{N}_1^n \cup \ldots \cup \mathcal{N}_r^n, \\ r_a &= 0, \quad \text{for } a \in \mathcal{N}_1^n \cup \ldots \cup \mathcal{N}_r^n, \\ C &= B, \\ B &= \mathrm{Id}. \end{aligned}$$

This is a place where a discontinuity (non-monotonicity) appears in the algorithm. A similar change of the Lipschitz part may be done when vectors r_a become thick in comparison to q_a.

6. Derivatives of the Poincaré map

Consider a differential equation

$$x' = f(x), \quad x \in \mathbb{R}^n, \quad f \in \mathcal{C}^{K+1}. \tag{37}$$

Let $\varphi : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ be a (local) dynamical system induced by (37). Let $\alpha : \mathbb{R}^n \to \mathbb{R}$ be a $\mathcal{C}^1$-map. Put $\Pi = \{x \mid \alpha(x) = C\}$.

Definition 6. *We will say that* Π *is a local section for the vector field* f *at* $y_0 \in \Pi$*, if*

$$\langle \nabla\alpha(y_0) | f(y_0) \rangle \neq 0. \tag{38}$$

Assume $x_0 \in \mathbb{R}^n$ and $t_0 \in \mathbb{R}$ are such that Π is a local section at $\varphi(t_0, x_0)$. Consider an implicit equation

$$\alpha(\varphi(t_P(x), x)) = C. \tag{39}$$

It follows easily from (38) and from the implicit function theorem that there exists a uniquely defined $t_P : \mathbb{R}^n \multimap \mathbb{R}$ in a neighborhood of x_0, such that $t_P(x_0) = t_0$. The function t_P is as smooth as the flow φ. We will refer to t_P as to the *Poincaré return time to section* Π.

We define a Poincaré map $P : \mathbb{R}^n \supset \mathrm{dom}\,(t_P) \to \mathbb{R}^n$ by

$$P(x) = \varphi(t_P(x), x). \tag{40}$$

Usually the Poincaré map is defined as a map $P : \Pi_1 \multimap \Pi_2$, where Π_1, Π_2 are some local sections in $\mathbb{R}^n$. The approach taken here, i.e. treating the Poincaré map as map $P : \mathbb{R}^n \multimap \mathbb{R}^n$, allows us not to worry about the coordinates on the local section.

We are interested in the partial derivatives of P defined by (40).
From (40) we obtain

$$\frac{\partial P_i}{\partial x_j}(x) = f_i(P(x))\frac{\partial t_P}{\partial x_j}(x) + \frac{\partial \varphi_i}{\partial x_j}(t_P(x), x). \tag{41}$$

We need $\frac{\partial t_P}{\partial x_j}$. We differentiate (39) to obtain

$$\sum_{k=1}^{n} \frac{\partial \alpha}{\partial x_k}(P(x)) \left(f_k(P(x))\frac{\partial t_P}{\partial x_j}(x) + \frac{\partial \varphi_k}{\partial x_j}(t_P(x), x) \right) = 0,$$

$$\langle \nabla\alpha(P(x)) | f(P(x)) \rangle \frac{\partial t_P}{\partial x_j}(x) + \sum_{k=1}^{n} \frac{\partial \alpha}{\partial x_k}(P(x))\frac{\partial \varphi_k}{\partial x_j}(t_P(x), x) = 0. \tag{42}$$

Hence,

$$\frac{\partial t_P}{\partial x_j}(x) = -\frac{1}{\langle \nabla\alpha(P(x)) | f(P(x)) \rangle} \sum_{k=1}^{n} \frac{\partial \alpha}{\partial x_k}(P(x))\frac{\partial \varphi_k}{\partial x_j}(t_P(x), x). \tag{43}$$

6.1. Higher order derivatives of the Poincaré map

To make the formulae transparent we will drop the arguments of the functions in this section, but reader should be aware that for t_P and its partial derivatives the argument is x, for φ and $D_a\varphi$ the argument is always the pair $(t_P(x), x)$.

From (41) we obtain

$$\begin{aligned} D_{(j,c)}P &= \frac{\partial^2}{\partial t^2}\varphi D_{(j)}t_P D_{(c)}t_P + \frac{\partial}{\partial t}D_{(c)}\varphi D_{(j)}t_P + \frac{\partial}{\partial t}\varphi D_{(j,c)}t_P \\ &+ \frac{\partial}{\partial t}D_{(j)}\varphi D_{(c)}t_P + D_{(j,c)}\varphi. \end{aligned}$$

It is easy to see that partial derivatives of the high order give rise to quite complex expressions and it is not entirely obvious how to organize it in some coherent and programmable way. For this purpose we use the following

Lemma 5. *For a multipointer $a \in \mathcal{N}_p^n$ we have*

$$\begin{aligned} D_aP &= D_a\varphi + \tfrac{\partial \varphi}{\partial t} D_a t_P \\ &+ \textstyle\sum_{k=2}^{p} \frac{\partial^k \varphi}{\partial t^k} \sum_{(\delta_1,\ldots,\delta_k)\in\mathcal{N}^p(k)} \prod_{j=1}^{k} D_{a_{\delta_j}} t_P \\ &+ \textstyle\sum_{k=2}^{p} \sum_{(\delta_1,\ldots,\delta_k)\in\mathcal{N}^p(k)} \sum_{s=1}^{k} \frac{\partial^{k-1}}{\partial t^{k-1}} D_{a_{\delta_s}}\varphi \prod_{j\neq s} D_{a_{\delta_j}} t_P. \end{aligned} \tag{44}$$

Proof: By induction on p. For $p = 1$ Eq. (44) is equivalent to (41), because the last two sums are taken over the empty set. Assume (44) holds true for some $p \geq 1$ and fix $a \in \mathcal{N}_{p+1}^n$. Our goal is to show that

$$D_aP = R_1 + R_2 + R_3,$$

where

$$\begin{aligned}
R_1 &= D_a\varphi + \frac{\partial}{\partial t}\varphi D_a t_P,\\
R_2 &= \sum_{k=2}^{p+1}\frac{\partial^k}{\partial t^k}\varphi \sum_{(\delta_1,\dots,\delta_k)\in\mathcal{N}^{p+1}(k)}\prod_{j=1}^{k} D_{a_{\delta_j}} t_P,\\
R_3 &= \sum_{k=2}^{p+1}\sum_{(\delta_1,\dots,\delta_k)\in\mathcal{N}^{p+1}(k)}\sum_{s=1}^{k}\frac{\partial^{k-1}}{\partial t^{k-1}} D_{a_{\delta_s}}\varphi\prod_{j\neq s} D_{a_{\delta_j}} t_P.
\end{aligned}$$

Write $a=\beta+\gamma$, where $\beta\in\mathcal{N}_p^n$ and $\gamma=(a_{p+1})\in\mathcal{N}_1^n$. From the induction assumption we have

$$\begin{aligned}
D_aP &= D_\gamma\left(D_\beta\varphi+\tfrac{\partial}{\partial t}\varphi D_\beta t_P\right)\\
&+ D_\gamma\left(\textstyle\sum_{k=2}^{p}\frac{\partial^k}{\partial t^k}\varphi\sum_{(\delta_1,\dots,\delta_k)\in\mathcal{N}^p(k)}\prod_{j=1}^{k} D_{\beta_{\delta_j}} t_P\right)\\
&+ D_\gamma\left(\textstyle\sum_{k=2}^{p}\sum_{(\delta_1,\dots,\delta_k)\in\mathcal{N}^p(k)}\sum_{s=1}^{k}\frac{\partial^{k-1}}{\partial t^{k-1}} D_{\beta_{\delta_s}}\varphi\prod_{j\neq s} D_{\beta_{\delta_j}} t_P\right)\\
&= \textstyle\sum_{i=1}^{10} S_i,
\end{aligned}$$

where

$$\begin{aligned}
S_1 &= D_a\varphi+\tfrac{\partial}{\partial t}\varphi D_a t_P,\\
S_2 &= \tfrac{\partial}{\partial t} D_\beta\varphi D_\gamma t_P,\\
S_3 &= \tfrac{\partial^2}{\partial t^2}\varphi D_\beta t_P D_\gamma t_P,\\
S_4 &= \tfrac{\partial}{\partial t} D_\gamma\varphi D_\beta t_P,\\
S_5 &= \textstyle\sum_{k=2}^{p}\frac{\partial^k}{\partial t^k} D_\gamma\varphi\sum_{(\delta_1,\dots,\delta_k)\in\mathcal{N}^p(k)}\prod_{j=1}^{k} D_{\beta_{\delta_j}} t_P,\\
S_6 &= \textstyle\sum_{k=2}^{p}\frac{\partial^{k+1}}{\partial t^{k+1}}\varphi D_\gamma t_P\sum_{(\delta_1,\dots,\delta_k)\in\mathcal{N}^p(k)}\prod_{j=1}^{k} D_{\beta_{\delta_j}} t_P,\\
S_7 &= \textstyle\sum_{k=2}^{p}\frac{\partial^k}{\partial t^k}\varphi\sum_{(\delta_1,\dots,\delta_k)\in\mathcal{N}^p(k)}\sum_{s=1}^{k} D_{\beta_{\delta_s}+\gamma} t_P\prod_{\substack{j=1\\ j\neq s}}^{k} D_{\beta_{\delta_j}} t_P,\\
S_8 &= \textstyle\sum_{k=2}^{p}\sum_{(\delta_1,\dots,\delta_k)\in\mathcal{N}^p(k)}\sum_{s=1}^{k}\frac{\partial^{k-1}}{\partial t^{k-1}} D_{\beta_{\delta_s}+\gamma}\varphi\prod_{j\neq s} D_{\beta_{\delta_j}} t_P,\\
S_9 &= \textstyle\sum_{k=2}^{p}\sum_{(\delta_1,\dots,\delta_k)\in\mathcal{N}^p(k)}\sum_{s=1}^{k}\frac{\partial^{k}}{\partial t^{k}} D_{\beta_{\delta_s}}\varphi D_\gamma t_P\prod_{j\neq s} D_{\beta_{\delta_j}} t_P,\\
S_{10} &= \textstyle\sum_{k=2}^{p}\sum_{(\delta_1,\dots,\delta_k)\in\mathcal{N}^p(k)}\\
&\quad \textstyle\sum_{s=1}^{k}\sum_{\substack{r=1\\ r\neq s}}^{k}\frac{\partial^{k-1}}{\partial t^{k-1}} D_{\beta_{\delta_s}}\varphi D_{\beta_{\delta_r}+\gamma} t_P\prod_{\substack{j\neq s\\ j\neq r}} D_{\beta_{\delta_j}} t_P.
\end{aligned}$$

Obviously $R_1=S_1$. We will show that $R_2=S_3+S_6+S_7$ and $R_3=S_2+S_4+S_5+S_8+S_9+S_{10}$.

Denote by $R_{i,k}$, $i=2,3$ a part of sum R_i with fixed $k=2,\dots,p+1$. Similarly, let us denote by $S_{i,k}$ a part of sum S_i, $i=5,\dots,10$, for $k=2,\dots,p$.

Using decomposition of $\mathcal{N}^{p+1}(2)$ as in (9) we obtain that $R_{2,2}=S_3+S_{7,2}$. Similarly, using (9) we observe that $R_{2,k}=S_{6,k-1}+S_{7,k}$ for $k=3,\dots,p$. Finally, since $\mathcal{N}^{p+1}(p+1)=\{((1),(2),\dots,(p+1))\}$ and $\gamma=(a_{p+1})$ we find that $R_{2,p+1}=S_{6,p}$. This shows that $R_2=S_3+S_6+S_7$.

It remains to show that $R_3=S_2+S_4+S_5+S_8+S_9+S_{10}$. We will classify possible terms by the place of the appearance of $p+1$ in δ_i, $i=1,\dots,k$ and by how this δ_i enters in R_3 as δ_s or δ_j. There are four cases:

1. $\delta_s=(p+1)$,

2. $\delta_j = (p+1)$,

3. $p+1 \in \delta_s$, $|\delta_s| \geq 2$,

4. $p+1 \in \delta_j$, $|\delta_j| \geq 2$.

Let us fix $k = 2$. Let $(\delta_1, \delta_2) \in \mathcal{N}^{p+1}(2)$. The term for case 1 is S_4, for case 2 is S_2, for case 3 is $S_{8,2}$ and for case 4 is $S_{10,2}$. Hence, $R_{3,2} = S_2 + S_4 + S_{8,2} + S_{10,2}$.

For $k = 3, \ldots, p$ and fixed $(\delta_1, \ldots, \delta_k) \in \mathcal{N}^{p+1}(k)$ case 1 is given by $S_{5,k-1}$, case 2 by $S_{9,k-1}$, case 3 by $S_{8,k}$ and case 4 by $S_{10,k}$ Hence, for $k = 3, \ldots, p$ we have $R_{3,k} = S_{5,k-1} + S_{9,k-1} + S_{8,k} + S_{10,k}$.

Finally, for $k = p+1$ we observe, that $R_{3,p+1} = S_{5,p} + S_{9,p}$. Indeed, in this case $(\delta_1, \ldots, \delta_{p+1}) = ((1), (2), \ldots, (p+1))$. Hence, for $\delta_s = \gamma$ we have term $S_{5,p}$ and for $\delta_s \neq \gamma$ we have $S_{9,p}$.

We have shown that $R_3 = S_2 + S_4 + S_5 + S_8 + S_9 + S_{10}$ and the proof is finished. ∎

Hence, if we know all the partial derivatives of t_P up to order p we can compute the partial derivatives of the Poincaré map up to the same order. In the next subsection we show how to compute the partial derivatives of t_P for affine sections.

6.2. Partial derivatives of t_P for affine sections

Assume $\alpha : \mathbb{R}^n \to \mathbb{R}$ is an affine map given by

$$\alpha(x) = \alpha_0 + \sum_{i=1}^{n} \alpha_i x_i.$$

This is a quite restrictive assumption about sections, but it leads to relatively simple formulae for $D_a t_P$ and it is sufficient for the applications we have in mind.

LEMMA 6. *For a multipointer $a \in \mathcal{N}_p^n$ holds*

$$\begin{aligned} -D_a t_P \left\langle \nabla\alpha | \frac{\partial}{\partial t} \varphi \right\rangle = \\ \langle \nabla\alpha | D_a \varphi \rangle + \sum_{k=2}^{p} \left\langle \nabla\alpha | \frac{\partial^k}{\partial t^k} \varphi \right\rangle \sum_{(\delta_1,\ldots,\delta_k) \in \mathcal{N}^p(k)} \prod_{j=1}^{k} D_{a_{\delta_j}} t_P \\ + \sum_{k=2}^{p} \sum_{(\delta_1,\ldots,\delta_k) \in \mathcal{N}^p(k)} \sum_{s=1}^{k} \left\langle \nabla\alpha | \frac{\partial^{k-1}}{\partial t^{k-1}} D_{a_{\delta_s}} \varphi \right\rangle \prod_{j \neq s} D_{a_{\delta_j}} t_P. \end{aligned}$$

Proof: The proof is a direct consequence of Lemma 5 and (39). Since α is affine, by differentiating of $\alpha(P(x)) = C$ we get $\langle \nabla\alpha | D_a P \rangle = 0$. Using formula (44) for $D_a P$ we obtain our assertion. ∎

Fix $[x] \subset \mathbb{R}^n$ and assume we have a rigorous bound for $t_P([x]) \in [t_1, t_2]$ (see [31, Section 6] for more details on this). Lemmas 6 and 5 show that given rigorous bounds

for the partial derivatives $D_a\varphi([t_1,t_2],[x])$ and $\frac{\partial^k}{\partial t^k}D_a\varphi([t_1,t_2],[x])$ up to some order p we can compute recursively the rigorous bounds for the partial derivatives of $t_P([x])$ and $P([x])$ up to the same order. Notice, that $\frac{\partial^k}{\partial t^k}D_a\varphi$ are given by Taylor coefficients of the solution of (16) with initial conditions $P([x])$ for the $\mathcal{C}^0$ part and $D_a\varphi(t_P(x),[x])$ for equations for variations. Hence, these coefficients can be easily computed using the automatic differentiation algorithm.

7. Comparison to the $\mathcal{C}^0$-solver

In this section we present results of comparison of the $\mathcal{C}^0$-solver applied to the second order variational equations with the $\mathcal{C}^2$-solver. We performed tests of these algorithms on some classical low dimensional examples, such as the Volterra-Lotka system

$$\begin{cases} \dot{x} = x(2-y), \\ \dot{y} = y(x-3), \end{cases} \tag{45}$$

the pendulum equation

$$\ddot{x} = -\sin(x), \tag{46}$$

the Lorenz system

$$\begin{cases} \dot{x} = 10(-x+y), \\ \dot{y} = 28x - y - xz, \\ \dot{z} = xy - \frac{8}{3}z, \end{cases} \tag{47}$$

the Michelson system

$$\dddot{x} + \dot{x} + \frac{1}{2}x^2 = 1, \tag{48}$$

the Rössler system

$$\begin{cases} \dot{x} = -(y+z), \\ \dot{y} = x + 0.2y, \\ \dot{z} = 0.2 + z(x-5.7), \end{cases} \tag{49}$$

and for the Hénon-Heiles system (Hamiltonian equation)

$$\begin{cases} \ddot{x} = -x - 2xy, \\ \ddot{y} = y^2 - y - x^2. \end{cases} \tag{50}$$

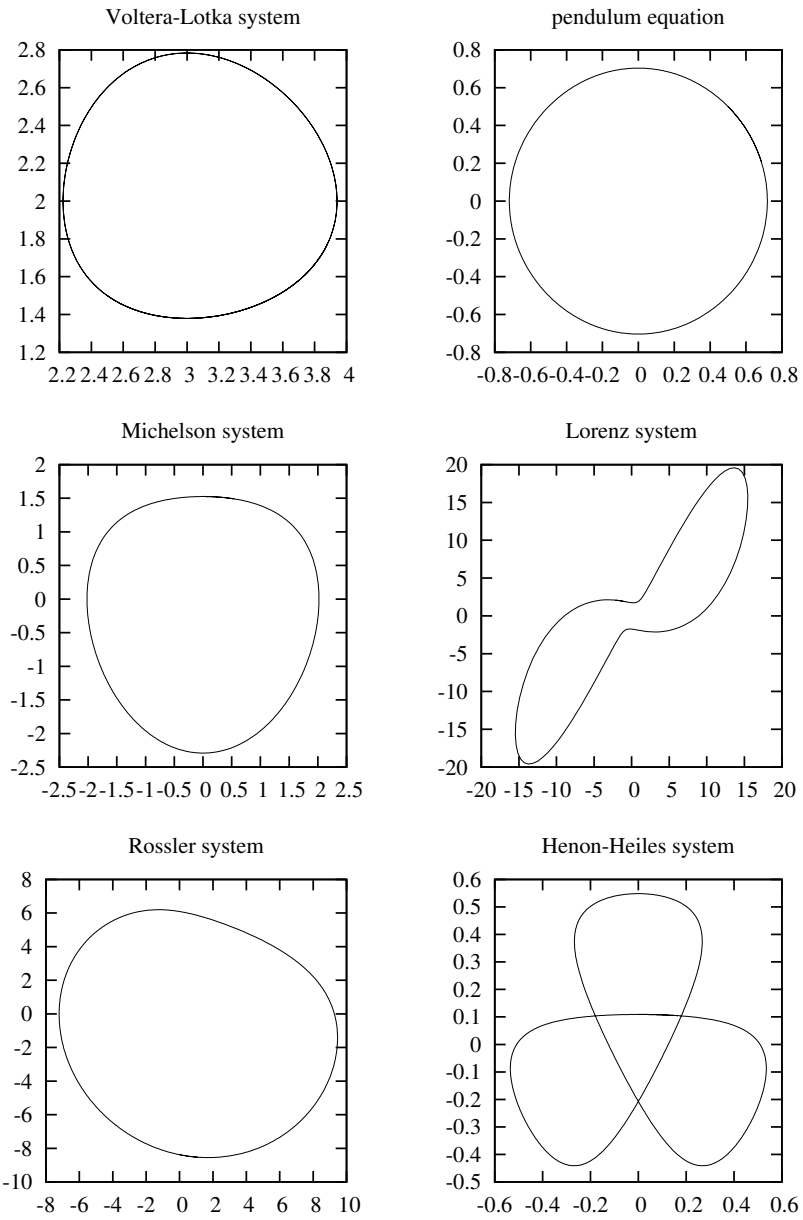

Fig. 1. Periodic orbits for the systems (45-50). The initial conditions are: $(2.5, 1.5)$ for the Volterra-Lotka system, $(0.5, 0.5)$ for the pendulum equation, $(0, 1.52596, 0)$ for the Michelson system, $(-2.14737, 2.07805, 27)$ for the Lorenz system, $(0., -8.3809417428298, 0.029590060630665)$ for the Rössler system and $(x, y, \dot{x}, \dot{y}) = (0, 0.10903, 0, 0.567723)$ for the Hénon-Heiles system

General settings of the tests.

- We integrate the above systems together with second and third order variational equations along periodic orbits using $\mathcal{C}^2$, $\mathcal{C}^3$ and $\mathcal{C}^0$ solvers from the CAPD library [4] to obtain bounds for the higher order derivatives. These periodic orbits are presented in Fig. 1. In each case the integration time is equal to an approximate period of the orbit. We believe that this is a relevant time scale for the computer assisted proofs for these systems.

- When integrating the systems of variational equations using the $\mathcal{C}^0$ solver we simply add the variational equations to the main equations and apply the $\mathcal{C}^0$ solver to the extended system that has dimension $n \begin{pmatrix} n+k \\ k \end{pmatrix}$, where n is the dimension of the main problem and k is the order of the derivatives we require.

- For each ODE (45)–(50) we set as initial conditions to each routine three boxes of diameters 0, 10^{-10} and 10^{-6} centered at a point very close to the corresponding periodic solution. The actual initial conditions are given in the caption of Fig. 1.

- In each case we use the Taylor method of order 20 with the variable time step. The minimal acceptable time step has been set to 10^{-5}. The computations were performed using the interval arithmetic with `double` precision.

Comparison of the computation times. As it is expected the $\mathcal{C}^2$ and $\mathcal{C}^3$ solvers are much faster than $\mathcal{C}^0$ applied to the equations for variations. In Tab. 1 we present the computation time (in seconds) for each problem when computed from a point initial condition ($\mathrm{diam}(x_0) = 0$). For 2–3 dimensional systems the speed up of the computation of second order derivatives was between 16 and 126. For the third order derivatives it is even larger and varies between 41 and 464.

For the Hénon-Heiles Hamiltonian the $\mathcal{C}^0$-solver was not able to integrate along the periodic solution neither second nor third order derivatives even when starting from a point initial condition. In Table 1 we gathered the computation times up to the blow-up which occurred at $t = 8.32874$ for the second order derivatives and $t = 3.6712$ for the third order derivatives. The total time of integration for this system has been set to $T = 13$.

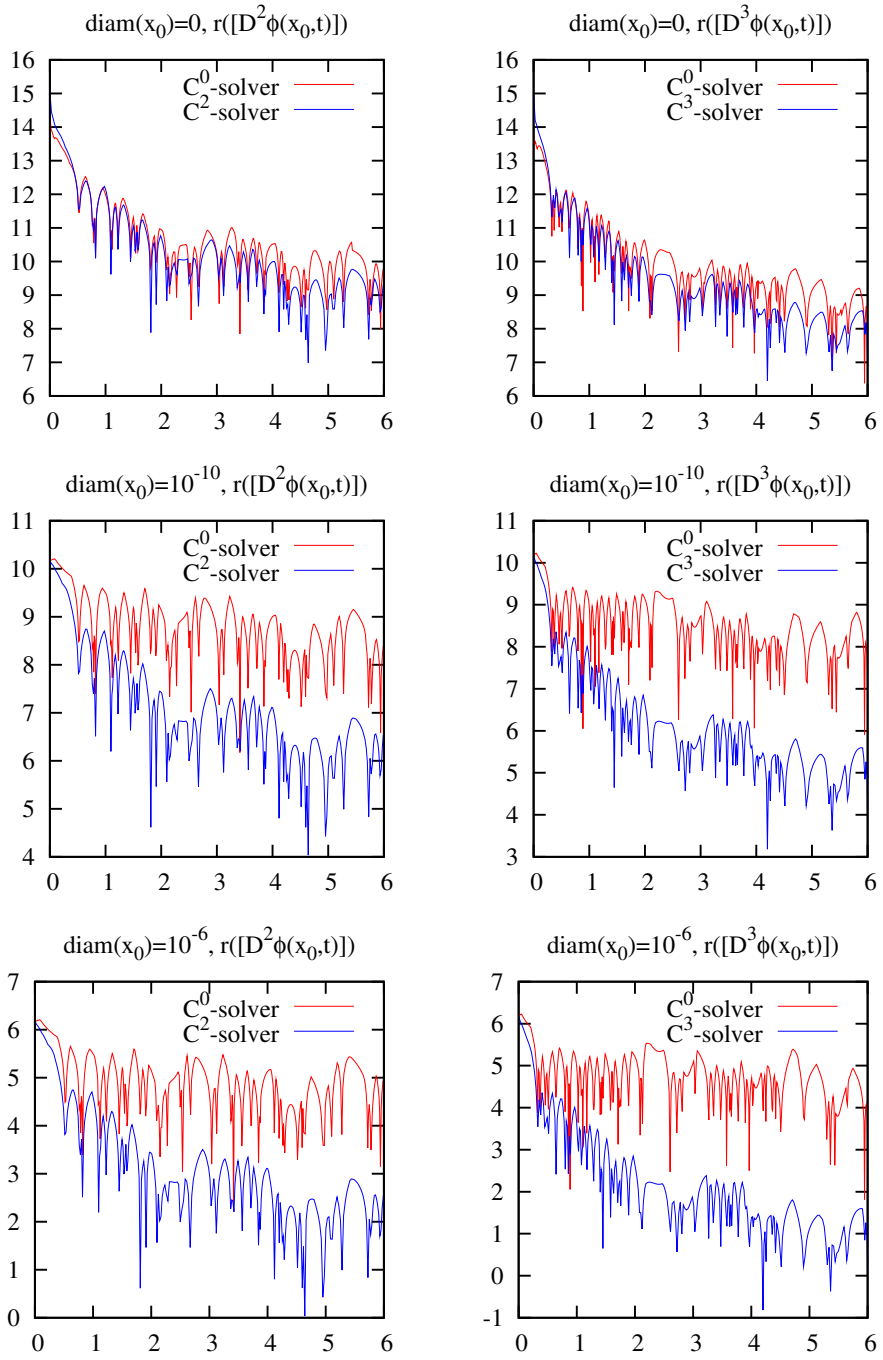

Fig. 2. Plots of $t \to r\left([D^2\phi(x_0,t)]\right)$ and $t \to r\left([D^3\phi(x_0,t)]\right)$ for the Volterra-Lotka system (45) obtained from $\mathcal{C}^0$ and $\mathcal{C}^r$ solvers for various diameters of initial conditions

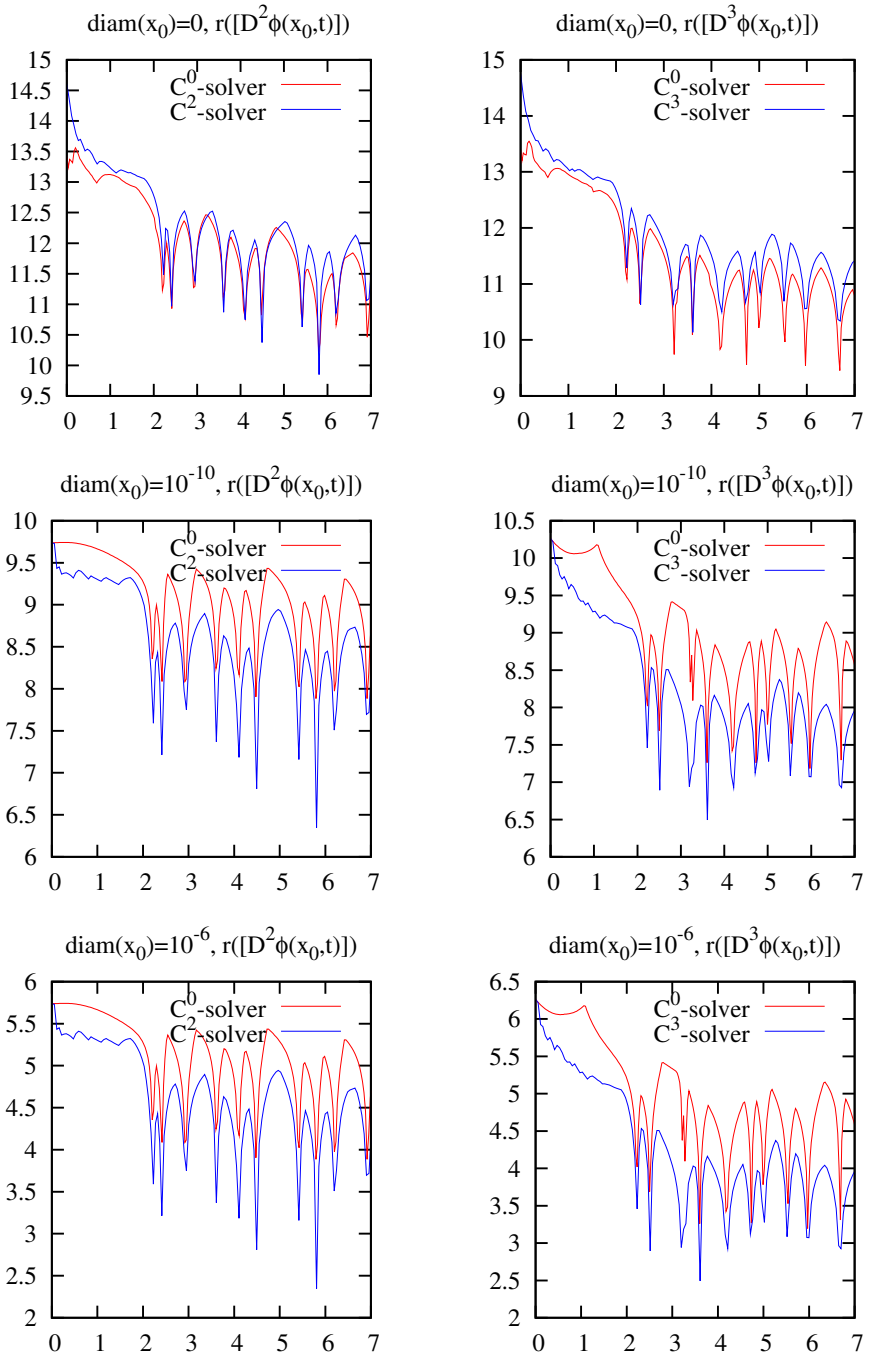

Fig. 3. Plots of $t \to r\left([D^2\phi(x_0,t)]\right)$ and $t \to r\left([D^3\phi(x_0,t)]\right)$ for the pendulum equation (46) obtained from $\mathcal{C}^0$ and $\mathcal{C}^r$ solvers for various diameters of initial conditions

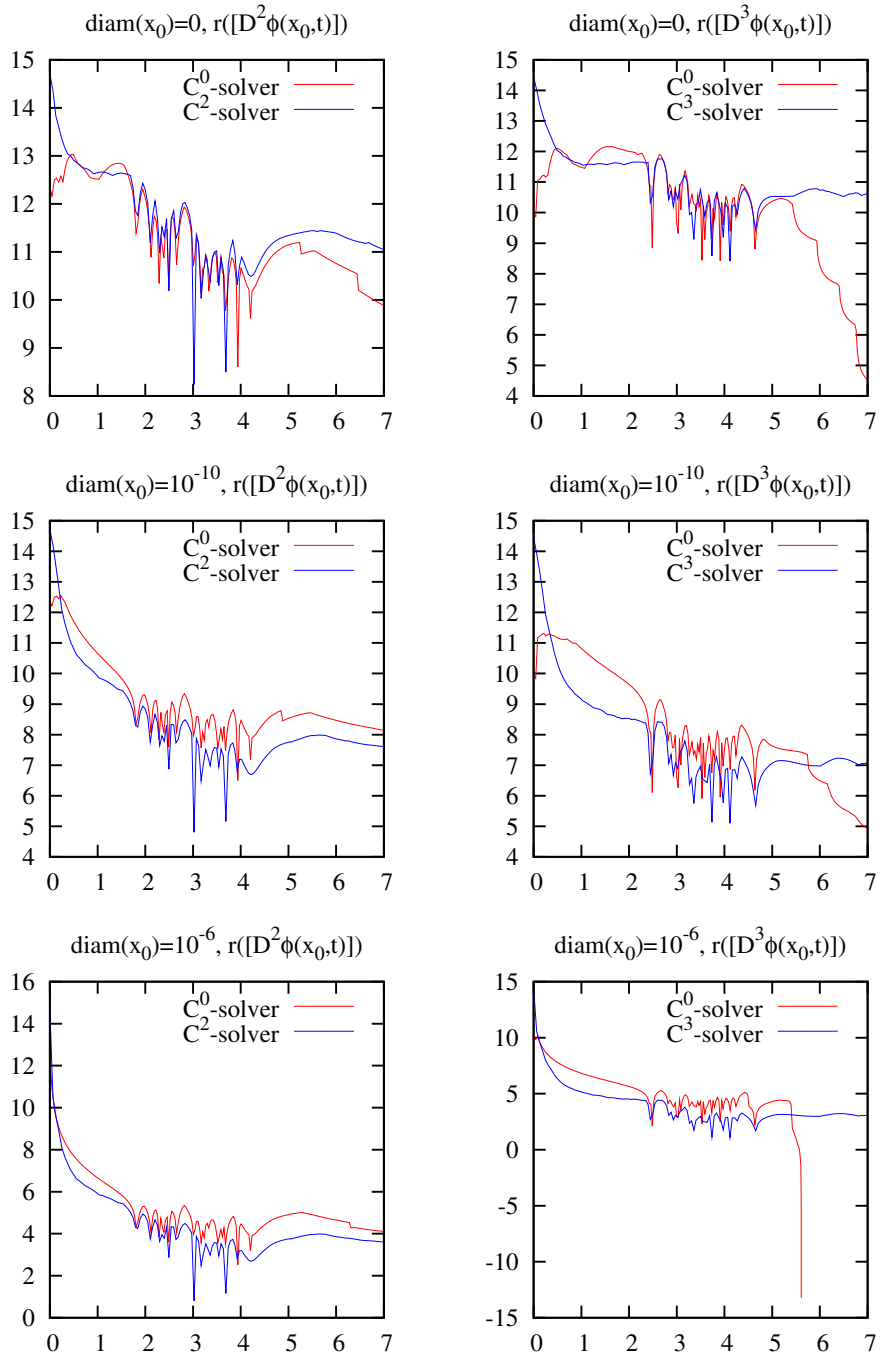

Fig. 4. Plots of $t \to r\left([D^2\phi(x_0,t)]\right)$ and $t \to r\left([D^3\phi(x_0,t)]\right)$ for the Michelson system (48) obtained from $\mathcal{C}^0$ and $\mathcal{C}^r$ solvers for various diameters of initial conditions

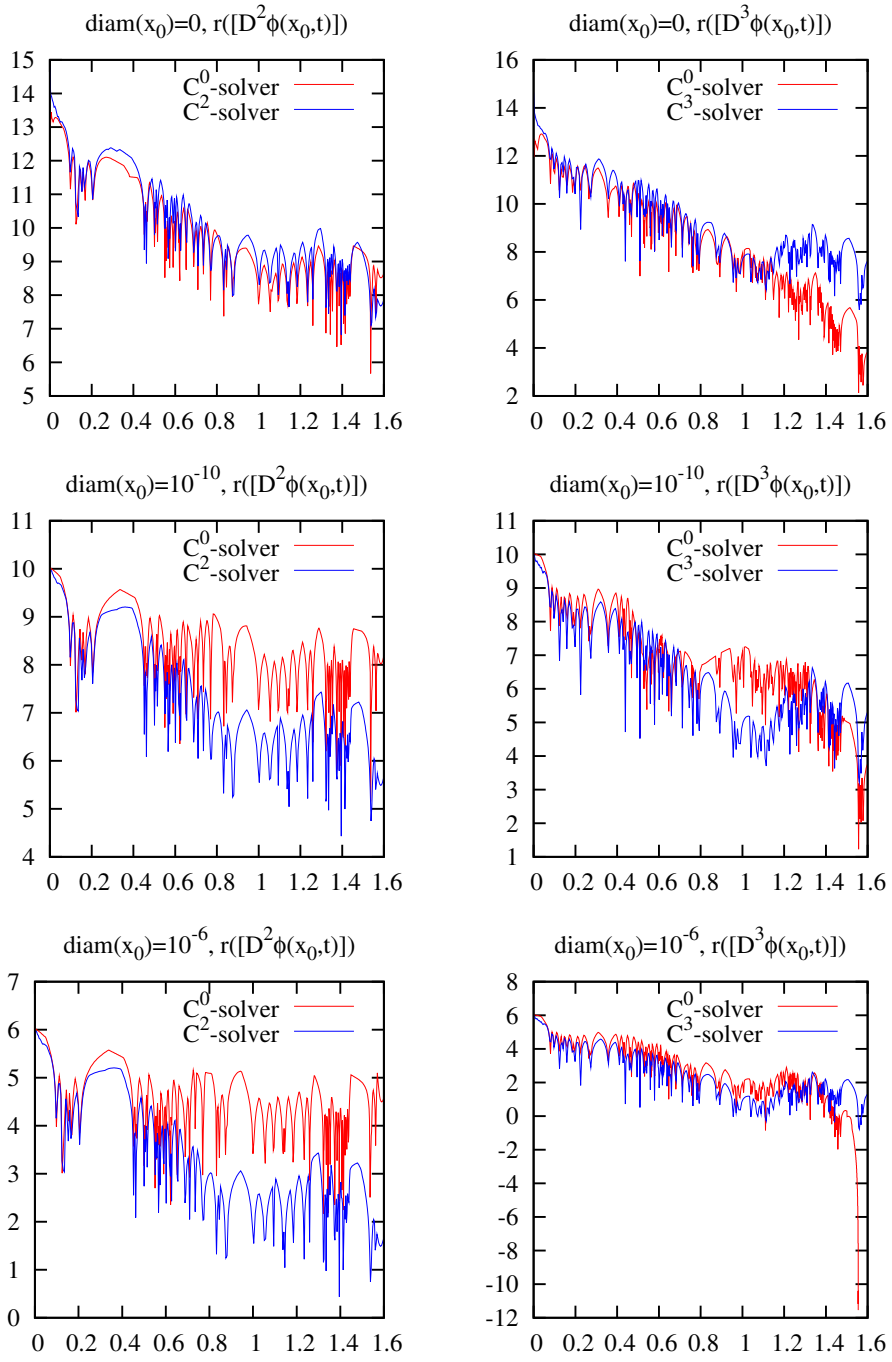

Fig. 5. Plots of $t \to r\left([D^2\phi(x_0,t)]\right)$ and $t \to r\left([D^3\phi(x_0,t)]\right)$ for the Lorenz system (47) obtained from $\mathcal{C}^0$ and $\mathcal{C}^r$ solvers for various diameters of initial conditions

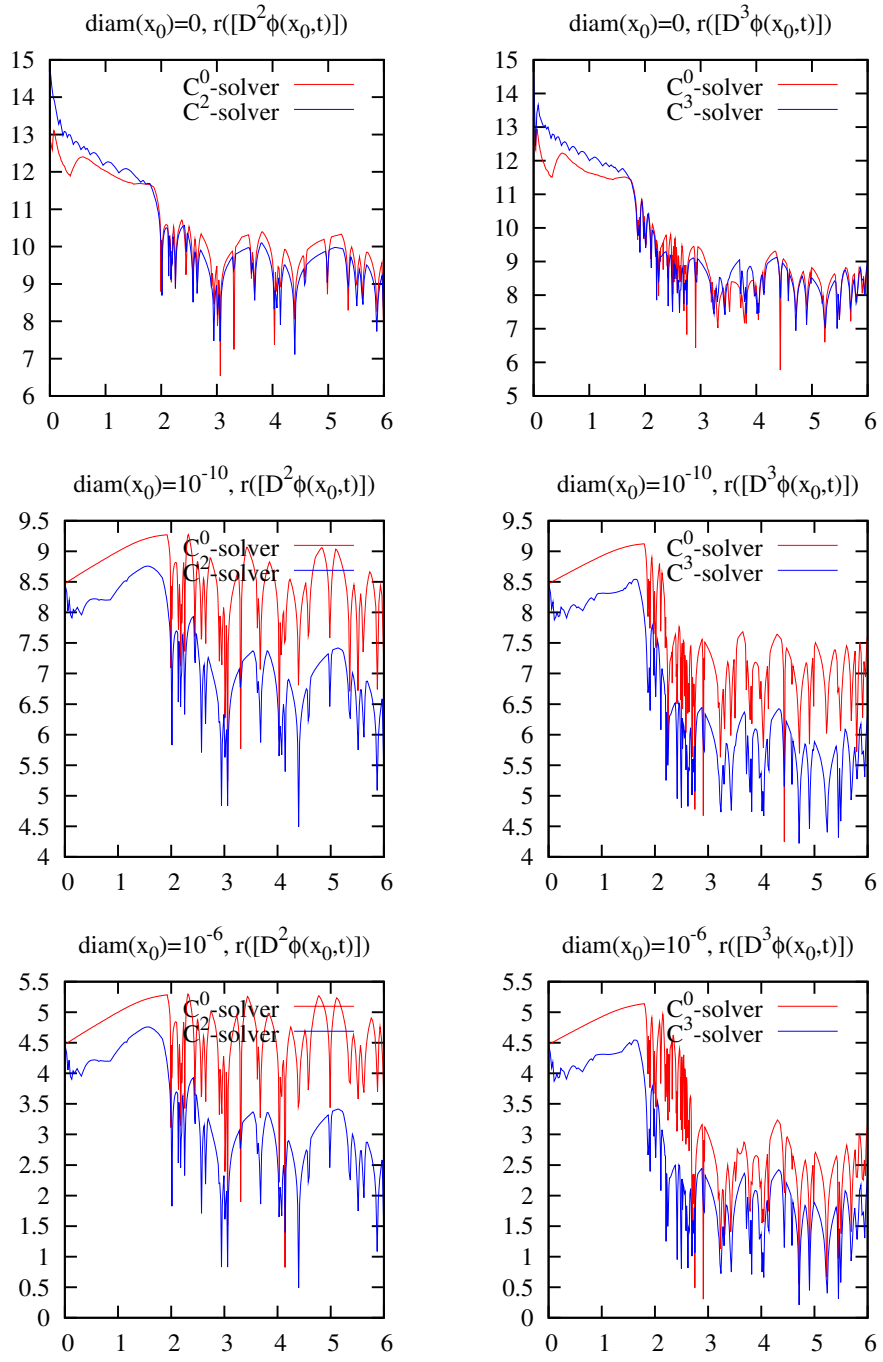

Fig. 6. Plots of $t \to r\left([D^2\phi(x_0,t)]\right)$ and $t \to r\left([D^3\phi(x_0,t)]\right)$ for the Rössler system (49) obtained from $\mathcal{C}^0$ and $\mathcal{C}^r$ solvers for various diameters of initial conditions

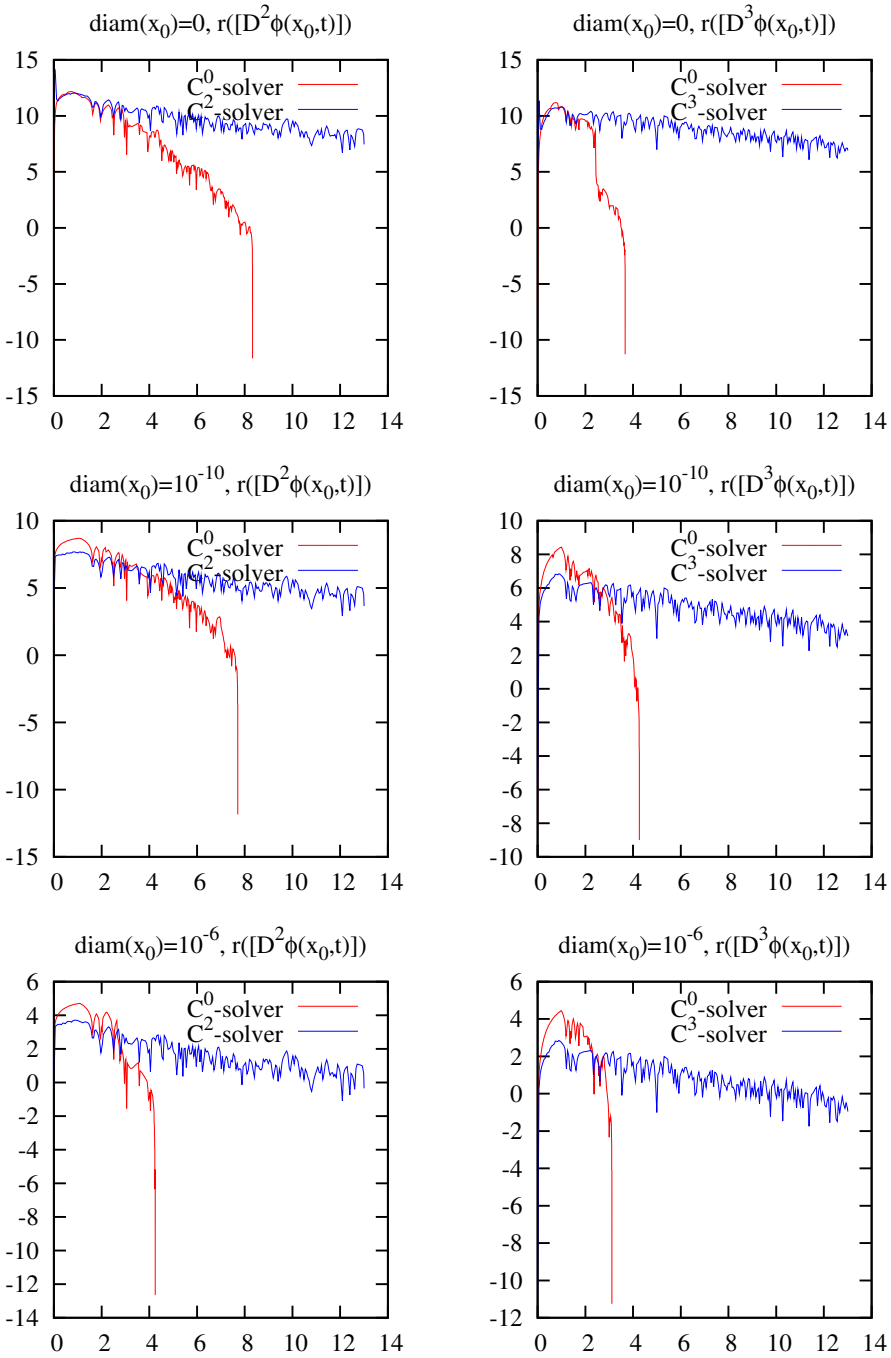

Fig. 7. Plots of $t \to r\left([D^2\phi(x_0,t)]\right)$ and $t \to r\left([D^3\phi(x_0,t)]\right)$ for the Hénon-Heiles system (50) obtained from $\mathcal{C}^0$ and $\mathcal{C}^r$ solvers for various diameters of initial conditions

Tab. 1. Comparison of the computation times for $\mathcal{C}^2$, $\mathcal{C}^3$ and $\mathcal{C}^0$ solvers when applied to the equations for variations. All computation times are given in seconds

The system	Second order derivatives			Third order derivatives		
	$\mathcal{C}^2$-solver	$\mathcal{C}^0$-solver	ratio	$\mathcal{C}^3$-solver	$\mathcal{C}^0$-solver	ratio
V-L	0.30	4.89	16	0.61	25.35	41
Pendulum	0.09	1.96	21	0.21	9.71	46
Michelson	0.20	25.30	126	0.51	237.14	464
Lorenz	1.22	81.59	66	3.48	762.34	219
Rössler	0.71	58.96	83	1.87	521.37	278
H-H	1.40	430.21	–	4.96	3001.63	–

Comparison of the obtained enclosures. For an interval $x = [a, b]$ we define a function

$$r(x) = -\log_{10}\left(\frac{b-a}{|\mathrm{mid}(x)|}\right) = -\log_{10}\left(\frac{2(b-a)}{|a+b|}\right).$$

For an interval $x = [a, b]$ that does not contain zero, the function r measures a relative diameter of x, i.e. an approximate number of significant decimal digits that are the same for a and b.

With some abuse of notation we will denote by the same letter a relative diameter of an interval vector $[u] \subset \mathbb{R}^m$, i.e.

$$r([u]) = \min\{r([u]_i) : i = 1, \ldots, m\}$$

and of an enclosure of k-th order derivative of a smooth function f

$$r\left([D^k f(x)]\right) = \min\{r\left([D_a f(x)]\right) : |a| = k\}.$$

In Figs 2–7 we present the plots of the relative diameters of $r\left(D^2\phi(x_0, t)\right)$ and $r\left(D^3\phi(x_0, t)\right)$ as a function of time t obtained from $\mathcal{C}^0$ and $\mathcal{C}^r$ solvers for various diameters of initial conditions for systems (45–50). Here ϕ denotes the local flow induced by the equation under consideration.

In principle, our $\mathcal{C}^r$-algorithm may be less accurate than the $\mathcal{C}^0$-Lohner direct solver in the computation of $D_a\varphi$ for $|a| \geq 1$, because we do not make use of the dependence of $D_a\varphi$ on x. Indeed, this can be seen for lower dimensional systems. But we have paid for this with the serious increase of the computation times.

For point initial conditions this lack of accuracy can be compensated by switching to the multiprecision arithmetic. In fact, for the systems under consideration we were able to obtain much thinner enclosures for derivatives using higher precision and within comparable or better time of computations.

For the initial conditions of nonzero diameters one can subdivide the sets. In many cases this strategy allows us to obtain better accuracy within the same or better time. In some cases, like the Volterra-Lotka system (45) and the Lorenz system (47) obtained enclosures are significantly better when integrating the variational

equations using the $\mathcal{C}^0$ solver. For these systems and low order derivatives one can choose between $\mathcal{C}^0$ and $\mathcal{C}^r$ solvers depending on the required accuracy of the result.

On the other hand, the $\mathcal{C}^0$ solver was not able to integrate third order derivatives for the Lorenz and Michelson systems when $\mathrm{diam}(x_0) = 10^{-6}$.

For higher dimensional systems, like the Hénon-Heiles Hamiltonian we see that the $\mathcal{C}^0$-solver cannot compete with $\mathcal{C}^r$-algorithm. Both, the time of computation and obtained enclosures for second and third order derivatives are worse than those resulting from the $\mathcal{C}^r$-solver.

Memory usage. We would like to mention that the direct $\mathcal{C}^0$-solver, when applied to the equations for variations, requires also a huge memory. This is due to the fact, that the $\mathcal{C}^0$ solver extended by the k-th order variational equations builds a tree for automatic differentiation for the system in $n \begin{pmatrix} n+k \\ k \end{pmatrix}$-dimensional space and also its derivative. This squares the effective dimension for the $\mathcal{C}^0$ solver. For the Hénon-Heiles system (50) and the third order derivatives the $\mathcal{C}^0$ solver used 22.31MB of RAM while the $\mathcal{C}^3$ solver used 416kB only.

Conclusions. The proposed algorithm has been proved to be very useful in many applications [12, 25, 28, 29]. In these papers we applied our $\mathcal{C}^2$–$\mathcal{C}^5$-algorithms to study various kinds of dynamic and bifurcations of ODEs. In all of these applications the desired accuracy of computed derivatives was not that large as we usually require for the $\mathcal{C}^0$ image of the set – only a very few significant digits were necessary to get the result. Our tests show that the $\mathcal{C}^r$ solver can compute high order derivatives with acceptable accuracy in a very good CPU time.

Our tests show also, that when the high accuracy of derivatives is required, the $\mathcal{C}^0$ solver applied to the equations for variations can compete with $\mathcal{C}^r$ solvers for low dimensional systems and for low order derivatives, only. This is due to the following facts:

- loss of control of the wrapping effect in the $\mathcal{C}^0$ solver when the dimension is really high,
- memory usage; for example, using our $\mathcal{C}^5$ solver we integrated along a periodic solution the fifth order derivatives of a Hamiltonian flow (n-body problem) in 8 dimensions. The program used 7GB of RAM. We were not able to build the $\mathcal{C}^0$ solver for the fifth order derivatives on a computer with 64GB of memory,
- even if possible to build the necessary objects in the memory, the time of computations for large problems would be very large.

8. References

[1] Alefeld G.; *Inclusion methods for systems of nonlinear equations – the interval Newton method and modifications*, in: Herzberger J. (ed.), *Topics in Validated Computations*, Elsevier Science B.V., 1994, pp. 7–26.

[2] Broer H.W., Huitema G.B., Sevryuk M.B.; *Quasi-periodicity in families of dynamical systems: order amidst chaos*, Lecture Notes in Mathematics, 1645, Springer Verlag, 1996.

[3] Berz M., Makino K.; *New Methods for High-Dimensional Verified Quadrature*, Reliable Computing, 5, 1999, pp. 13–22.

[4] CAPD – Computer Assisted Proofs in Dynamics group; a C++ package for rigorous numerics. Available via `http://capd.wsb-nlu.edu.pl`.

[5] Griewank A.; *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, Frontiers in Applied Mathematics, 19, SIAM, 2000.

[6] Galias Z., Zgliczyński P.; *Computer assisted proof of chaos in the Lorenz system*, Physica D, 115(3–4), 1998, pp. 165–188.

[7] Jorba À., Zou M.; *A software package for the numerical integration of ODE by means of high-order Taylor methods*, Experimental Mathematics, 14, 2005, pp. 99–117.

[8] Hardy M.; *Combinatorics of Partial Derivatives*, Electronic Journal of Combinatorics, 13, 2006.

[9] Hairer E., Nørsett S.P., Wanner G.; *Solving Ordinary Differential Equations I, Nonstiff Problems*, Springer-Verlag, Berlin–Heidelberg, 1987.

[10] Hassard B., Zhang J., Hastings S., Troy W.; *A computer proof that the Lorenz equations have "chaotic" solutions*, Applied Mathematics Letters, 7, 1994, pp. 79–83.

[11] Kapela T., Zgliczyński P.; *The existence of simple choreographies for N-body problem – a computer assisted proof*, Nonlinearity, 16, 2003, pp. 1899–1918.

[12] Kokubu H., Wilczak D., Zgliczyński P.; *Rigorous verification of cocoon bifurcations in the Michelson system*, Nonlinearity, 20, 2007, pp. 2147–2174.

[13] Lohner R.J.; *Computation of Guaranteed Enclosures for the Solutions of Ordinary Initial and Boundary Value Problems*, in: Cash J.R., Gladwell I. (ed.), *Computational Ordinary Differential Equations*, Clarendon Press, Oxford 1992.

[14] Michelson D.; *Steady solutions of the Kuramoto–Sivashinsky equation*, Physica D, 19, 1986, pp. 89–111.

[15] Moore R.E.; *Interval Analysis*, Prentice Hall, 1966.

[16] Mischaikow K., Mrozek M.; *Chaos in the Lorenz equations: A computer assisted proof*, Mathematics of Computation, 67, 1998, pp. 1023–1046.

[17] Mrozek M., Zgliczyński P.; *Set arithmetic and the enclosing problem in dynamics*, Annales Polonici Mathematici, 2000, pp. 237–259.

[18] Nedialkov N.S., Jackson K.R.; *An Interval Hermite – Obreschkoff Method for Computing Rigorous Bounds on the Solution of an Initial Value Problem for an Ordinary Differential Equation*, in: Csendes T. (ed.), *Developments in Reliable Computing*, Kluwer, Dordrecht, Netherlands, 1999, pp. 289–310.

[19] Neumeier A.; *Interval methods for systems of equations*, Cambridge University Press, 1990.

[20] Rall L.B.; *Automatic Differentiation: Techniques and Applications*, Lecture Notes in Computer Science, 120, 1981.

[21] Rage T., Neumaier A., Schlier C.; *Rigorous verification of chaos in a molecular model*, Phys. Rev. E, 50, 1994, pp. 2682–2688.

[22] Rössler O.E.; *An Equation for Continuous Chaos*, Physics Letters A, 57(5), 1976, pp. 397–398.

[23] Tucker W.; *A Rigorous ODE solver and Smale's 14th Problem*, Foundations of Computational Mathematics, 2(1), 2002, pp. 53–117.

[24] Walter W.; *Differential and integral inequalities*, Springer-Verlag, New York 1970.

[25] Wilczak D.; *Rigorous normal forms and the existence of KAM invariant curves for Poincaré maps*, in review.

[26] Wilczak D.; *Symmetric heteroclinic connections in the Michelson system – a computer assisted proof*, SIAM Journal on Applied Dynamical Systems, 4(3), 2005, pp. 489–514.

[27] Wilczak D., Zgliczyński P.; *Heteroclinic Connections between Periodic Orbits in Planar Restricted Circular Three Body Problem – A Computer Assisted Proof*, Communications in Mathematical Physics, 234, 2003, pp. 37–75.

[28] Wilczak D., Zgliczyński P.; *Computer assisted proof of the existence of homoclinic tangency for the Henon map and for the forced-damped pendulum*, SIAM Journal on Applied Dynamical Systems, 8(4), 2009, pp. 1632–1663.

[29] Wilczak D., Zgliczyński. P.; *Period doubling in the Rössler system – a computer assisted proof*, Foundations of Computational Mathematics, 9, 2009, pp. 611–649.

[30] Zgliczyński P.; *Computer assisted proof of chaos in the Hénon map and in the Rössler equations*, Nonlinearity, 10(1), 1997, 243–252.

[31] Zgliczyński P.; *C^1-Lohner algorithm*, Foundations of Computational Mathematics, 2, 2002, pp. 429–465.

Received March 18, 2010