J Sci Comput (2014) 59:28–52
DOI 10.1007/s10915-013-9749-1

Efficient and Generic Algorithm for Rigorous Integration Forward in Time of dPDEs: Part I

Jacek Cyranka

Received: 7 January 2013 / Revised: 26 June 2013 / Accepted: 2 July 2013 / Published online: 25 July 2013

Abstract We propose an efficient and generic algorithm for rigorous integration forward in time of partial differential equations written in the Fourier basis. By rigorous integration we mean a procedure which operates on sets and return sets which are guaranteed to contain the exact solution. The presented algorithm generates, in an efficient way, normalized derivatives which are used by the Lohner algorithm to produce a rigorous bound. The algorithm has been successfully tested on several partial differential equations (PDEs) including the Burgers equation, the Kuramoto-Sivashinsky equation, and the Swift-Hohenberg equation. The problem of rigorous integration in time of partial differential equations is a problem of large computational complexity and efficient algorithms are required to deal with PDEs on higher dimensional domains, like the Navier-Stokes equation. Technicalities regarding the various optimization techniques implemented in the software used in this paper will be reported elsewhere.

Keywords Dissipative PDE · Rigorous numerics · Automatic differentiation · Fast fourier transform · FFT · Interval arithmetic

Mathematics Subject Classification Primary: 65M99, 35B40 · Secondary: 35B41, 65G20, 65M70, 65Y20, 65T50

1 Introduction

The main goal of this paper is to present an efficient algorithm for rigorous integration of dissipative partial differential equations (dPDEs). By *rigorous integration* we mean a process in which a compact and connected set containing the unique solution is propagated along the time interval. The rigorous integration process guarantees to include the exact solution in the output set. Whereas in any standard (non-rigorous) integration process a

J. Cyranka (✉)
Institute of Computer Science, Jagiellonian University, prof. Stanisława Łojasiewicza 6, 30-348 Kraków, Poland
e-mail: jacek.cyranka@ii.uj.edu.pl

point, represented by floating point numbers, is being propagated without indication how close is the approximated solution to the real one.

Our main objective in the rigorous integration task is to perform computer assisted proofs of the existence of certain solutions: attractors, periodic orbits, connections between fixed points etc.

Sometimes, when one is willing to perform a computer assisted proof of the existence of a particular solution, the integration process can be avoided. Advantage of the methods which do not perform numerical integration is that the error involved in computing rigorously a solution using a boundary value setup is more distributed along the whole orbit. Examples of rigorous computational methods to study PDEs, different than the approach considered here—i.e. not involving the integration process include [3,9,10,15], and [25].

The main motivation behind this work was to reduce the actual execution time of a sequential algorithm for rigorous integration of dPDEs. We emphasize that the problem of rigorous integration of dPDEs is, by its nature, hard to parallelize, therefore any reduction of computational complexity is highly appreciated.

We have performed several rigorous numerics tests to show that

- the presented algorithm is apparently more efficient than the currently used algorithm for rigorous numerical integration of dPDEs used e.g. in [7] and [35],
- the presented algorithm produces negligibly worse bounds than the currently used algorithms.

The proposed approach in this paper combines several techniques. In this paper we briefly present issues arising when dealing with rigorous calculations and present methods which we used to circumvent the issues. We briefly present techniques used to obtain the algorithm generality, i.e. to make it easily adaptable to different equations and boundary conditions that could still be studied using Fourier series. More details and the proofs are presented in [8].

We restrict ourselves to the following problem

$$\begin{aligned} &u\colon [0,T)\times\mathbb{T}\to\mathbb{R},\quad f,u_0\colon \mathbb{T}\to\mathbb{R},\\ &u_t = L(u)+N(u,u_x,u_{xx},\dots)+f,\\ &u(0,x)=u_0(x), \end{aligned} \tag{1}$$

where L is a linear elliptic operator (for instance $L(u)=\nu u_{xx}$ or $L(u)=-\nu u_{xxxx}-u_{xx}$), $N(u,u_x,u_{xx},\dots)$ is a proper polynomial of u and its partial derivatives, f is a constant in time forcing, $\mathcal{T}$ is a maximal time of existence of the solution, $\mathbb{T}$ is the one-dimensional torus.

We restrict $N(u,u_x,u_{xx},\dots)$ to be a proper polynomial, as not any equation (1) is a dPDE. It is additionally required from (1) to be called a dPDE that the linear part must "dominate" in some sense the nonlinear part, for the formal definition refer to [35].

The Fourier basis is introduced and the problem of solving (1) is reduced to the problem of solving the following infinite dimensional dynamical system

$$\frac{da_k}{dt} = c_N \sum_{\substack{k_1,\dots,k_n\in\mathbb{Z}\\ k_1+\dots+k_n=k}} k_1^{r_1}a_{k_1}k_2^{r_2}a_{k_2}\dots k_n^{r_n}a_{k_n} + \lambda_k a_k + f_k,\quad k\in\mathbb{Z}, \tag{2}$$

where λ_k are eigenvalues of the elliptic operator, c_N, n, and r_j depend on $N(u,u_x,u_{xx},\dots)$, $\{a_k\}_{k\in\mathbb{Z}}\subset\mathbb{C}$ describes the evolution of the Fourier modes. Let us mention few important equations which fall into the dPDE class: the Burgers equation, the Kuramoto-Sivashinsky equation, the Ginzburg-Landau equation, the Swift-Hohenberg equation and, last but not least, the Navier-Stokes equation.

The system (2) is an infinite dimensional system of ordinary differential equations (ODEs), and rigorously integrating it on a computer requires special techniques. To the best of our knowledge, there exist two published theoretical approaches to the rigorous numerics solution for the non-stationary PDE problem—the method of self-consistent bounds, presented in the series of papers [32,34–36], and the method presented in [1].

Both of the mentioned methods have been successfully applied to the study of the Kuramoto-Sivashinsky equation dynamics (KS equation). In fact, the existence of some time-periodic solutions has been proved. The focus of the previous research on one particular equation comes from the interest in establishing the first proof of the existence of spatio-temporal chaotic dynamics in a partial differential equation.

The presented algorithm is a first attempt to provide a generic and robust tool (a computer software) for the rigorous analysis of any dPDE, not only the KS equation.

Moreover, the proposed algorithm improves the efficiency of the rigorous integration task, which is of great importance for us, as the attempt to prove the existence of chaotic dynamics in the KS equation by using our method will require performing a lot of proofs in parallel. We hope that the presented work will permit computer assisted proofs concerning the dynamics of higher dimensional dPDEs, including the Navier-Stokes equations. We consider especially interesting 3D Navier-Stokes equations, because for this case even the fundamental question of the existence and uniqueness of solutions has no definitive answer yet. It is worth noting that there already exist results in literature about the rigorous numerics for higher dimensional PDEs, see e.g. [13,20], and [22], but in this work the stationary problem is considered only, the equations are not being numerically integrated at any point.

The conclusion that we draw is that the algorithm is a proper approach to the task of the rigorous integration of higher dimensional dPDEs, including the Navier-Stokes equations, which we will address in our forthcoming work.

2 Fast Fourier Transform algorithm

Any attempt to integrate in time an equation of the type (2) will require a significant number of convolution calculations. It is well known that calculation of this type of convolutions is done efficiently by using *the Fast Fourier Transform (FFT) algorithm*. Fields of applicability of the FFT are, for instance, numerical solving of partial differential equations for weather prediction, a signal analysis and image processing in computer graphics. To the best of our knowledge, the FFT algorithm was described for the first time in the paper [6]. We refer to the papers [29] and [30] for an excellent review. In this case it is enough to know how to calculate a finite vector field approximately and the application is straight-forward. One would think that it is so in the case of rigorous methods too. In fact, in rigorous methods the situation is much more difficult. We address here the questions related to the application of the FFT algorithm within a rigorous method setting, including how the interval FFT algorithm behaves, whether the *wrapping effect* (see [33] for the statement of the wrapping effect problem) will be present, how to calculate the higher order time derivatives required by the Taylor method, and what optimizations to perform. We answer all these questions here, we introduce a new technique of eliminating the aliasing error, and in [8] we present the proofs and details. FFT was already used to study dPDEs by the rigorous numerics community in [11,12], and [16,17].

2.1 Outline of Approach

Notation From now on we assume that N is a positive natural number not to be confused with the polynomial $N(u, u_x, u_{xx}, \ldots)$ in (1).

When using a spectral method based on the Fourier series one often has to compute a considerable amount of convolutions of a finite sequence $\{u_k\}_{k=-N}^{N}$

$$(u * \cdots * u)_k = \sum_{\substack{k_1+k_2+k_3+\cdots+k_n=k \\ -N\le k_i\le N}} u_{k_1} u_{k_2} \ldots u_{k_n}, \qquad k = -N, \ldots, N, \tag{3}$$

where n is the number of terms in the convolution $u * \cdots * u$.

It is known in the numerical analysis community [5] that computation of (3) could be considerably accelerated by means of the FFT algorithm. The idea of the approach is based on the following facts

- the function $u(x) = \sum_{k=-N}^{N} u_k \exp(ikx)$ is in the unique way determined by $u(2\pi j/M)$, where $M \ge 2N + 1$ and $j = 0, 1, \ldots, M - 1$,
- passing from the values at the nodes $u(2\pi j/M)$, $j = 0, \ldots, M - 1$ to the Fourier coefficients u_k, $k = -N, \ldots, N$ and vice-versa is done efficiently by using the FFT algorithm.

Therefore a possibly effective approach of computation of (3) is the following

0. Assume that $u(x) = \sum_{j=-N}^{N} u_j \exp(ijx)$.
1. Choose big enough M, see the discussion of *the aliasing problem* in Sect. 3.1.
2. Let $x_j := \frac{2\pi j}{M}$, $j = 0, 1, \ldots, M - 1$. Compute $\hat{u}_j = u(x_j)$ using FFT.
3. Compute $\hat{q}_j = \hat{u}_j \ldots \hat{u}_j$ for $j = 0, 1, \ldots, M - 1$. Modulo *the aliasing error*, we have $(u \ldots u)(x_j) = \hat{q}_j$.
4. Using FFT applied to $\hat{q}_j$ we obtain the Fourier coefficients for u^n given by (3).

2.2 Uniform Matrix Form of the FFT

Definition 1 Let $v : \mathbb{R} \to \mathbb{C}$ be a 2π-periodic function such that v belongs to the subspace spanned by the functions $\{\exp(-iNx), \ldots, \exp(iNx)\}$, i.e. $v(x) = \sum_{k=-N}^{N} v_k \exp(ikx)$. Then we call

$$\{v_k\}_{k=-N}^{N}$$

the $l_2(N)$ *coefficients* of v. When it is clear from the context we will drop the notation of N, and use simply the l_2 coefficients of v.

Definition 2 Let $v : \mathbb{R} \to \mathbb{C}$ be a 2π-periodic function such that v belongs to the subspace spanned by the functions $\{\exp(-iNx), \ldots, \exp(iNx)\}$, i.e. $v(x) = \sum_{k=-N}^{N} v_k \exp(ikx)$. Then we call

$$v(x_j) = \hat{v}_j = \sum_{k=-N}^{N} v_k \exp(ikx_j), \quad j = 0, \ldots, M - 1$$

the $L_2(M)$ *coefficients* of v. When it is clear from context we will drop the notation of M, and use simply the L_2 coefficients of v.

By $l_2(N) \to L_2(M)$ transform (later on called $l_2 \to L_2$ transform) we mean simply the evaluation of $\{\hat{u}_j\}_0^{M-1}$ from $\{u_k\}_{-N}^{N}$, being in fact the Discrete Fourier Transform

$$\hat{u}_j = \sum_{k=-N}^{N} u_k \exp(ijk2\pi/M), \quad j = 0, \ldots, M-1. \tag{4}$$

From now on we assume $M \geq 2N+1$.

By $L_2(M) \to l_2(N)$ transform (later on called $L_2 \to l_2$ transform) we mean the computation of the Fourier coefficients

$$u_k = \frac{1}{M} \sum_{j=0}^{M-1} \hat{u}_j \exp(-ijk2\pi/M), \quad k = -N, \ldots, N. \tag{5}$$

The idea behind the FFT algorithm is that (4) and (5) are unified in a matrix form, and then the resulting matrices are written as a product of sparse matrices. Let us fix N and M. We map the l_2 coefficients $\{u_k\}_{k=-N}^{N}$ and the L_2 coefficients $\{\hat{u}_j\}_{j=0}^{M-1}$ to M dimensional vectors z and $\hat{z}$ in the following way

$$\begin{aligned} z_k &:= \begin{cases} u_k, & k = 0, \ldots, N, \\ u_{k-M}, & k = M-N, \ldots, M-1, \\ 0, & \text{otherwise} \end{cases} \\ \hat{z}_j &:= \hat{u}_j, \quad j = 0, \ldots, M-1. \end{aligned} \tag{6}$$

Then the matrix form is given by the following Lemma

Lemma 1 (the FFT matrix form) *Let $M \geq 2N+1$,*

$$\hat{z} = W_M z, \quad z = \overline{W_M}\hat{z}, \tag{7}$$

where $W_M(j,k) := \exp(ijk2\pi/M)$, $\overline{W_M}(j,k) := \frac{1}{M}\exp(-ijk2\pi/M)$.

To evaluate (7) efficiently one does not perform the multiplication by W_M, which is a dense matrix but one factorizes W_M into *sparse matrices* and perform the multiplication by sparse matrices.

3 Issues arising when the FFT algorithm is used within interval arithmetic framework

In this section we describe what the aliasing error is and we provide techniques that allow to eliminate it.

We emphasize that the technique presented in this section is apparently more efficient than the standard one, currently used in the rigorous numerics community, see [11] and references supplied there. Our approach is based on the technique introduced in [26]. The authors of [26] introduced new shifted discrete grids in order to eliminate the aliasing error in three dimensional convolutions arising in the Navier-Stokes equations. As we present here, shifted discrete grids can be applied to rigorous methods for dPDEs in 1D and apparently yield a substantial improvement.

3.1 Aliasing Problem

When the convolutions like (3) are calculated using the approach outlined in Sect. 2.1 the so-called aliasing problem appears.

Convolution of n terms We set

$$s_k := (u * \cdots * u)_k = \sum_{\substack{k_1+k_2+\cdots+k_n=k \\ -N\le k_j\le N}} u_{k_1}u_{k_2}\ldots u_{k_n}, \quad k=-N,\ldots,N.$$

Consider now the convolution of n terms, $u*\cdots*u$. Then the result of the multiplication of the L_2 coefficients is the following

$$\hat{s}_j = \hat{u}_j \ldots \hat{u}_j.$$

Observe that $\{\hat{s}_j\}_{j=0}^{M-1}$ are L_2-coefficients of the following function

$$x \mapsto \sum_{k=-nN}^{nN} \sum_{k_1+k_2+\cdots+k_n=k} u_{k_1}u_{k_2}\ldots u_{k_n}\exp(ikx).$$

$L_2 \to l_2$ transform of $\hat{s}_j$ gives

$$\tilde{s}_k := \frac{1}{M}\sum_{j=0}^{M-1} \hat{s}_j \exp(-ikj2\pi/M) = \sum_{\substack{a\in\mathbb{Z} \\ -nN\le k+aM\le nN}} \sum_{k_1+k_2+\cdots+k_n=k+aM} u_{k_1}\ldots u_{k_n} =$$

$$\sum_{k_1+k_2+\cdots+k_n=k} u_{k_1}\ldots u_{k_n} + \sum_{\substack{a\in\mathbb{Z}\setminus\{0\} \\ -nN\le k+aM\le nN}} \sum_{k_1+k_2+\cdots+k_n=k+aM} u_{k_1}\ldots u_{k_n}.$$

The term

$$\sum_{\substack{a\in\mathbb{Z}\setminus\{0\} \\ -nN\le k+aM\le nN}} \sum_{k_1+k_2+\cdots+k_n=k+aM} u_{k_1}\ldots u_{k_n}$$

is the aliasing error. The range of a appearing in the sum of the aliasing error is bounded, namely if $aM > (n+1)N$, then for all $k\in\{-N,\ldots,N-1,N\}$ holds $k+aM > nN$ and $k-aM < -nN$.

Definition 3 We call $\tilde{s}_k$'s as above *the aliased convolutions*.

3.2 Standard Technique of Aliasing Error Removal (Padding)

The aliasing error is eliminated simply by choosing the number of grid points M large enough. This is the approach used in [11] and other related works. The following condition has to be satisfied in the general case

$$k+M > nN \quad \forall k\in\{-N,\ldots,N\}, \text{ and } k-M < -nN \quad \forall k\in\{-N,\ldots,N\},$$

we take the worst case condition and obtain

$$N-M < -nN, \quad -N+M > nN,$$
$$M > (n+1)N.$$

Therefore when $M > (n+1)N$ the aliasing error is not present.

3.3 Technique Based on Phase Shifts

This technique is designed only for the functions, for which the Fourier coefficients $\{u_k\}$ are purely real or purely imaginary (for instance real odd or real even functions). To remove the aliasing error one may also use a phase shift, that is calculate L_2 coefficients at the points belonging to the uniform grid shifted to the right by a constant value Δ.

Let

$$\hat{u}_j^{\Delta} = \sum_{k=-N}^{N} u_k \exp(ik\,(\Delta + j2\pi/M)), \quad j = 0, \ldots, M-1,$$
$$\hat{s}_j^{\Delta} = \hat{u}_j^{\Delta} \ldots \hat{u}_j^{\Delta}.$$

The particular choice of $\Delta = \pi/(2M)$ is interesting for our purposes. The following lemma holds, a proof is provided in [8].

Lemma 2 *Let M be the number of the points on the grid. Let $u : \mathbb{R} \to \mathbb{C}$ be a 2π-periodic function such that u belongs to the subspace spanned by the functions $\{\exp(-iNx), \ldots, \exp(iNx)\}$ and, moreover, it is odd or even.*

Assume that $M > \frac{(n+1)}{2}N$, then $\mathrm{Re}\left(s_k^{\pi/(2M)}\right) = s_k$ (for u even or n even) or $\mathrm{Im}\left(s_k^{\pi/(2M)}\right) = s_k$ (for u odd and n odd), for all $-N \le k \le N$.

Now, let us put the presented techniques into the context of the FFT algorithm. Whatever aliasing elimination technique is used in the FFT approach to calculate convolutions, the $M \ge 2N+1$ relation has to be in any case satisfied to allow all modes (whose number is $2N+1$) to be mapped by the mapping (6).

To perform this technique approximately $\frac{(n+1)}{2}N \log \frac{(n+1)}{2}N$ operations are required, whereas to perform technique from 3.2 approximately $(n+1)N \log (n+1)N$.

Although the difference is asymptotically negligible, for the input sizes which are attainable in practical applications the number of operations in this technique compared to the technique from 3.2 is reduced by a significant factor. This is due to a reduction of the aliasing error influence in this technique—the values in the reduced number of nodes are calculated.

3.4 Overestimates

As we are interested in rigorous computations, therefore all computations have to be performed using *the interval arithmetic* [24,31]. In our implementation we used the infimum-supremum representation of intervals. In this subsection we address the question if the rigorous evaluation of the convolutions (3) using the FFT algorithm produces more overestimates than the direct evaluation. We will refer to *the interval FFT algorithm* as *the rigorous FFT algorithm*.

We have not found a comprehensive study of the overestimates that arise in the rigorous FFT algorithm in the existing literature. We have performed numerical tests in order to provide clear answer to the question if the rigorous evaluation of the convolutions (3) using the FFT algorithm produces more overestimates compared to the direct evaluation.

In the test presented in this section the function u has been a real-valued and odd function belonging to the space spanned by a finite number of vectors, randomly picked for each single test, all u_k have been intervals. We have fixed $C > 0, s > 0$ and required that $\{u_k\}_{k=-N}^{N}$ satisfy $\mathrm{mid}\,[u_k] \in [-C/k^s, C/k^s]$, in order to mimic regular functions. The diameter (width) was the same for all u_k.

Table 1 Tests for $N = 21$, $s_k := \sum_{k_1+k_2=k} u_{k_1} u_{k_2}$

u_k diameter(u_k width)	Maximum over coordinates of diameters s_k (maximal width of s_k)		
	Direct evaluation	Algorithm FFT	
		I padding	II phase shift
		$M = 72 = 4 \cdot 3 \cdot 3 \cdot 2$	$M = 45 = 5 \cdot 3 \cdot 3$
1e−05	1.76984e−05	0.00045232	0.00121946
0.0001	0.000176984	0.0045232	0.0121946
0.001	0.00176984	0.045232	0.121946
0.01	0.0184213	0.46557	1.42349
0.1	0.340996	18.8056	82.8143
1	21.8849	1677.01	7732.67
2	85.7698	6662.83	30808.7
5	529.425	41473.1	192097

Table 2 Tests for $N = 21$, $s_k := \sum_{k_1+k_2+k_3=k} u_{k_1} u_{k_2} u_{k_3}$

u_k diameter (u_k width)	Maximum over coordinates of diameters s_k (maximal width of s_k)		
	Direct eval	Algorithm FFT	
		I padding	II phase shift
		$M = 90 = 5 \cdot 3 \cdot 3 \cdot 2$	$M = 45 = 5 \cdot 3 \cdot 3$
1e−05	6.87813e−05	0.00176142	0.00297904
0.0001	0.000687813	0.0176144	0.0297908
0.001	0.00688878	0.176266	0.298266
0.01	0.0729481	1.88594	3.33923
0.1	1.70614	165.085	430.783
1	406.774	121698	343497
2	2883.3	956535	2.71295e+06
5	42922	1.48829e+07	4.225e+07

In each test we used the techniques presented in Sects. 3.2 and 3.3 in order to eliminate the aliasing error. We present example results in Tables 1, 2, and 3.

Apparently, the rigorous FFT algorithm produces more overestimates, especially when sets of large diameter (of order > 1) are used. The considerably larger overestimates produced by the FFT compared to the direct method are probably due to several matrix-vector multiplications performed by the FFT algorithm, it is known that matrix-vector multiplications are the main source of the so-called wrapping effect, for precise statement of the wrapping effect refer e.g. [23], and [33].

4 A Rigorous dPDE Integrator

To explain what the term *rigorous integration* mean let us consider the following abstract Cauchy problem for a system of ordinary differential equations (ODEs)

Table 3 Tests for $N = 21$, $s_k := \sum_{k_1+k_2+k_3+k_4+k_5=k} u_{k_1} u_{k_2} u_{k_3} u_{k_4} u_{k_5}$

u_k diameter (u_k width)	Maximum over coordinates of diameters s_k (maximal width of s_k)		
	Direct eval	Algorithm FFT	
		I padding	II phase shift
		$M = 144 = 4 \cdot 4 \cdot 3 \cdot 3$	$M = 72 = 4 \cdot 3 \cdot 3 \cdot 2$
1e−05	7.15846e−05	0.000709162	0.001178
0.0001	0.000715878	0.00709179	0.0117803
0.001	0.00718731	0.0710905	0.118113
0.01	0.752179	10.4391	18.3479
0.1	36.2275	1696.94	2861.64
1	184009	5.98891e+07	8.89027e+07
2	4.63014e+06	1.80823e+09	2.65626e+09
5	3.8902e+08	1.70558e+11	2.48932e+11

$$\begin{cases} \dot{x}(t) = f(x(t)), \\ x(0) = \quad x_0. \end{cases} \tag{8}$$

$x\colon [0, \mathcal{T}) \to \mathbb{R}^n$, $f\colon \mathbb{R}^n \to \mathbb{R}^n$, $f \in C^\infty$. The goal of a rigorous ODEs solver is to find a compact and connected set $\mathbf{x_k} \subset \mathbb{R}^n$ such that

$$\varphi(t_k, \mathbf{x_0}) \subset \mathbf{x_k}, \tag{9}$$

$t_k \in [0, \mathcal{T})$, $\mathbf{x_0} \subset \mathbb{R}^n$. By $\varphi(t_k, x_0)$ we denote the solution of (8) at the time t_k with initial condition $x_0 \in \mathbb{R}^n$, and therefore $\varphi(t_k, \mathbf{x_0})$ denotes the set of all the values which are attained at the time t_k by any solution of (8) with the initial condition in $\mathbf{x_0}$.

Notation We denote by $[x]$ an *interval set* $[x] \subset \mathbb{R}^n$, $[x] = \Pi_{k=1}^n [x_k^-, x_k^+]$, $[x_k^-, x_k^+] \subset \mathbb{R}$, $-\infty < x_k^- \le x_k^+ < \infty$, mid $([x])$ is the middle of an interval set $[x]$. There exist a few algorithms that offer reliable computations of the solution trajectories for ODEs that are based on interval arithmetic. The approach used in this paper is based on the Lohner algorithm, introduced in [21], see also [33]. To obtain a rigorous bound for the solution and avoid at the same time the so-called wrapping effect we calculate a bound for the partial derivative with respect to the initial conditions. Basically, in the Lohner algorithm (see e.g. [33]) instead of calculating directly the expression

$$\Phi_q(h, [x]) + R_{\Phi_q}([W]) \supset \varphi(h, [x]),$$

the mean value form is used

$$\Phi_q(h, \text{mid}\,([x])) + \partial \Phi_q / \partial x\,(h, [x])\,(\text{mid}\,([x]) - [x]) + R_{\Phi_q}([W]) \supset \varphi(h, [x]), \tag{10}$$

$h > 0$ is a time step, Φ_q is a q-th order numerical method (for the purpose of this paper, *the Taylor method*) for an equation (8), R_{Φ_q} is a remainder term calculated using a convex hull of an enclosure for the solution values in the time interval $[0, h]$, i.e. a set $[W] \subset \mathbb{R}^n$ such that $\varphi([0, h], [x]) \subset [W]$.

Bounds for the set

$$\varphi(h, [x]), \tag{11}$$

and bounds for the set containing partial derivatives with respect to the initial condition

$$\partial \Phi_q / \partial x \, (h, [x]) \tag{12}$$

are both obtained by the Taylor method in our approach. To obtain bounds for (12) we will use a suitably modified Taylor method. Here we do not present in detail the Taylor method and the Lohner algorithm and we refer the reader to [21,27] and [33].

Obtaining Taylor coefficients of any order Now, we address the question of how to obtain bounds for the Taylor coefficients of any order efficiently.

The algorithms from [33] used some explicit formulas for the Taylor coefficients of any order. These explicit formulas were derived for the second degree polynomials only. This approach was used in the following work considering rigorous dPDE integration, i.e. [32,34–36]. We found this approach awkward, mainly because of its troublesome implementation and limited applicability. Apparently, this approach works only for dPDEs with the non-linear term being a second degree polynomial.

Instead, we found a more suitable to use algorithm which combines *the jet transport* and *the automatic differentiation* techniques. These techniques have already proven to be extremely useful in many applications including a long-time integration of the solar system (see [18] and references given therein) or computer assisted proofs for ODEs.

4.1 Automatic Differentiation

Now, we address the question of how to obtain the derivatives with respect to time of a sufficiently smooth function efficiently. Here we present a convenient and yet efficient approach called *the automatic differentiation*, see e.g. [14,24]. *The automatic differentiation* has many applications beside of our interest. It is a general technique of obtaining recursively a fixed number of normalized derivatives of a general operation/procedure.

Definition 4 Let $n > 0$, $a\colon [0, t_{max}) \to \mathbb{R}$ be a sufficiently smooth function, we call

$$a^{[n]}(t) = \frac{1}{n!} a^{(n)}(t)$$

its n-th normalized derivative.

Let $b, c\colon [0, t_{max}) \to \mathbb{R}$ be sufficiently smooth functions, assume that $a(t) = G(b(t), c(t))$. Using the automatic differentiation the normalized derivatives of $a(t)$ can be obtained assuming that G is a composition of some basic operations for which the recursive evaluation formulas are known. Recursive formulas for basic operations (such as multiplication, exponential, logarithm, trigonometric functions etc.) are easily derived, see for example [18].

Only particular systems originating from a dPDE with a polynomial non-linearity are in the scope of this paper. Rules for the addition and the multiplication of two functions are required only, which will became apparent later on. The scalar multiplication, being trivial, is not presented here. Assume that $b^{[j]}, c^{[j]}$ are known for $j = 0, \ldots, r$ then

$$\text{if } a(t) = b(t) + c(t) \text{ then } a^{[r]}(t) = b^{[r]}(t) + c^{[r]}(t), \tag{13}$$

$$\text{if } a(t) = b(t) \cdot c(t) \text{ then } a^{[r]}(t) = \sum_{j=0}^{r} b^{[j]}(t) \cdot c^{[r-j]}(t). \tag{14}$$

Now let us consider a differential equation

$$\frac{dx}{dt} = F(x), \quad x \in \mathbb{R}^n, \quad F : \mathbb{R}^n \to \mathbb{R}^n. \tag{15}$$

Given $x^{[0]} = x^{(0)}$ we calculate $x^{[j]}$ for $j = 1, \ldots, q-1$, by the following recursive formula

$$x^{[j+1]} = \frac{1}{j+1} (F \circ x)^{[j]},$$

where $(F \circ x)^{[j]}$ is the j-th normalized derivative of the right-hand side function of (15) (F composed with x as a function of t).

4.2 Jet Transport

Now, we address the question how to obtain bounds for the partial derivative with respect to the initial condition (12). The most convenient approach is to modify the automatic differentiation procedure to make it calculate (11) and (12) simultaneously. This is done by the *jet transport* technique.

Definition 5 Let $n > 0$ be a fixed integer, $\check{a} \in \mathbb{R}$, $a_j \in \mathbb{R}$, $j = 0, \ldots, n-1$. We call a first degree polynomial of n variables $\boldsymbol{\xi} = (\xi_0, \xi_1, \ldots, \xi_{n-1})$

$$a(\boldsymbol{\xi}) := \check{a} + \sum_{j=0}^{n-1} a_j \xi_j$$

the first order jet.

An algebra of the first order jets is formed with the addition and multiplication operations defined as follows

Definition 6 Let $n > 0$, $a(\boldsymbol{\xi}), b(\boldsymbol{\xi})$ be first order jets, $\boldsymbol{\xi} = (\xi_0, \ldots, \xi_{n-1})$ are symbols. We define the addition $+$ of first order jets as follows

$$a(\boldsymbol{\xi}) + b(\boldsymbol{\xi}) := \check{a} + \check{b} + \sum_{j=0}^{n-1} \left(a_j + b_j\right) \xi_j.$$

We define the multiplication $\cdot$ of first order jets as follows

$$a(\boldsymbol{\xi}) \cdot b(\boldsymbol{\xi}) := \check{a} \cdot \check{b} + \sum_{j=0}^{n-1} \left(\check{a} \cdot b_j + \check{b} \cdot a_j\right) \xi_j.$$

Definition 7 Let $f : \mathbb{R}^n \to \mathbb{R}$, $t \geq 0$ be a time, $x_0 \in \mathbb{R}^n$ be a fixed value, $\boldsymbol{\xi} = (\xi_0, \ldots, \xi_{n-1})$ are symbols. We define the following first order jet for the function f

$$J_f(\boldsymbol{\xi}) := f(x_0) + \sum_{l=0}^{n-1} \frac{\partial f}{\partial x_l}(x_0) \xi_l.$$

The definition above has a straightforward extension to the case when f is vector-valued. When f is vector-valued, the vector of jets for f is denoted by $\mathbf{J_f}(\boldsymbol{\xi})$. Using the definition above we fix $t \geq 0$, $x_0 \in \mathbb{R}^n$, define the first order jets for $\varphi(t, x_0)$ and calculate

$$\varphi^{[j]}(t, x_0), \qquad j = 1, \ldots, q-1$$

using the automatic differentiation procedure described in Sect. 4.1, where $\varphi(t, x_0)$ is the smooth solution of (15). Precisely

$$\varphi^{[j]}(t, x_0) := \frac{1}{j!}\frac{\partial^j \varphi(t, x_0)}{\partial t^i}, \quad \left(\frac{\partial \varphi}{\partial x}\right)^{[j]}(t, x_0) := \frac{1}{j!}\frac{\partial^j}{\partial t^i}\frac{\partial \varphi(t, x_0)}{\partial x}, \quad j = 1, \ldots, q-1.$$

In what follows in addition to the regular Taylor method (Φ_q), we are going to use an extended variant (Φ_q^{jet}) whose coefficients are $\left\{\mathbf{J}_{\varphi^{[0]}}(\boldsymbol{\xi}), \ldots, \mathbf{J}_{\varphi^{[q-1]}}(\boldsymbol{\xi})\right\}$.

Φ_q^{jet} works as follows: the base part of $\mathbf{J}_{\varphi^{[0]}}(\boldsymbol{\xi})$ is set to the value of the current initial condition and the variational part of $\mathbf{J}_{\varphi^{[0]}}(\boldsymbol{\xi})$ is set to be the identity. The higher order coefficients are calculated performing the same operations as in the regular method, but on first order jets instead of scalars, and operations are performed with the rules of the algebra defined in Definition 6. This allows to solve (15) and the variational equations corresponding to it simultaneously.

4.3 An Algorithm Combining Automatic Differentiation with FFT

In this section we propose an algorithm for calculating both the solution of systems of ODEs originating from a dPDE and a matrix of partial derivatives with respect to the initial condition.

We emphasize the fact that the output of the presented algorithm is not enough to integrate efficiently a given system forward in time. This is not a stand-alone algorithm, nonetheless the output of the presented algorithm is necessary for any effective rigorous solver based on the Taylor method. Later we have combined it with the Lohner algorithm [21] in order to integrate example systems forward in time rigorously and we investigate how the techniques interplay with each other and what is the quality of the results.

The algorithm has been designed for dPDEs written using the truncated Fourier basis. We consider the following abstract system

$$\frac{da_k}{dt} = N_k(a) + \lambda_k a_k = F_k(a), \quad k = -N, \ldots, N. \tag{16}$$

Notation $a := \{a_k\}_{k=-N}^{N} \in \mathbb{C}^{2N+1}$ is a finite set of Fourier coefficients, $N_k(a)$ is a non-linear term comprising a convolution ($\sum_{k_1+k_2=k} a_{k_1} a_{k_2}$ for instance). Let $\varphi([0, h], [x_0]) \subset \mathbb{C}^{2N+1}$ denote all values of the solution within the time interval $[0, h]$, and starting with the initial condition in $[x_0] \subset \mathbb{C}^{2N+1}$.

From now on let φ be regular with respect to time solution of (16). Observe that the coefficients $\{a_k\}_{k=-N}^{N}$ are complex in this case, and $\varphi\colon [0, t_{max}) \times \mathbb{C}^{2N+1} \to \mathbb{C}^{2N+1}$.

First, let us comment on the possible approaches used to deal with the first order jets of complex functions. First possible approach is based on the following observation

Definition 8 Let $\{a_k\}_{k=-N}^{N} \in \mathbb{C}^{2N+1}$ satisfy $a_{-k} = \overline{a_k}$. We define $\Pi_{Re}\colon \mathbb{C}^{2N+1} \to \mathbb{R}^{2(N+1)}$ to be the following projection

$$\Pi_{Re}(a_{-N}, \ldots, a_N) = (\mathrm{Re}(a_0), \mathrm{Im}(a_0), \ldots, \mathrm{Re}(a_N), \mathrm{Im}(a_N)).$$

Observation 1 *Any function*

$$(a_{-N}, \ldots, a_N) \mapsto f(a_{-N}, \ldots, a_N),$$

can be translated to a function of real variables

$$(\mathrm{Re}\,(a_0)\,,\mathrm{Im}\,(a_0)\,,\ldots,\mathrm{Re}\,(a_N)\,,\mathrm{Im}\,(a_N)) \mapsto$$
$$\widetilde{f}\,(\mathrm{Re}\,(a_0)\,,\mathrm{Im}\,(a_0)\,,\ldots,\mathrm{Re}\,(a_N)\,,\mathrm{Im}\,(a_N))$$

in such a way that if $a_{-k} = \overline{a_k}$ *then*

$$f\,(a_{-N},\ldots,a_N) = \widetilde{f}\,(\Pi_{Re}\,(a_{-N},\ldots,a_N))\,.$$

By using Observation 1 we are able to translate our system (16) and use real first order jets in order to calculate the partial derivative with respect to the initial condition of the numerical method. But then a lot of information about the structure of the system is lost, and we do not recommend to use this approach.

The approach that we use is the following. Instead we use "complex" first order jets, which have the following form, let $(a_{-N},\ldots,a_N) \mapsto f\,(a_{-N},\ldots,a_N)$, $f\colon \mathbb{C}^{2N+1} \to \mathbb{C}$

$$J_f\,(\boldsymbol{\xi}) := f(x_0) + \sum_{l=-N}^{N} \begin{bmatrix} \frac{\partial\,\mathrm{Re}(f(x_0))}{\partial\,\mathrm{Re}(a_l)} + i\frac{\partial\,\mathrm{Im}(f(x_0))}{\partial\,\mathrm{Re}(a_l)} \\ \frac{\partial\,\mathrm{Re}(f(x_0))}{\partial\,\mathrm{Im}(a_l)} + i\frac{\partial\,\mathrm{Im}(f(x_0))}{\partial\,\mathrm{Im}(a_l)} \end{bmatrix}^T \xi_l. \tag{17}$$

We claim that using this kind of jets greatly facilitates computations and permits certain optimizations. This approach is strictly of computational interest. Below we provide a sketch of a justification of the presented approach.

Let $a(\xi)$ and $b(\xi)$ be arbitrary "complex" first order jets for two complex variables

$$a(\xi) = \check{c} + i\check{d} + \begin{bmatrix} c_1 + id_1 \\ c_2 + id_2 \end{bmatrix}^T \xi_1 + \begin{bmatrix} c_3 + id_3 \\ c_4 + id_4 \end{bmatrix}^T \xi_2\,,$$
$$b(\xi) = \check{e} + i\check{f} + \begin{bmatrix} e_1 + if_1 \\ e_2 + if_2 \end{bmatrix}^T \xi_1 + \begin{bmatrix} e_3 + if_3 \\ e_4 + if_4 \end{bmatrix}^T \xi_2\,.$$

Let us take the corresponding real first order jets $c = \mathrm{Re}\,(a)$, $d = \mathrm{Im}\,(a)$, $e = \mathrm{Re}\,(b)$ and $f = \mathrm{Im}\,(b)$. The corresponding first order jets are

$$\begin{aligned} c(\xi) &= \check{c} + c_1\xi_{1,1} + c_2\xi_{1,2} + c_3\xi_{2,1} + c_4\xi_{2,2}, \\ d(\xi) &= \check{d} + d_1\xi_{1,1} + d_2\xi_{1,2} + d_3\xi_{2,1} + d_4\xi_{2,2}, \\ e(\xi) &= \check{e} + e_1\xi_{1,1} + e_2\xi_{1,2} + e_3\xi_{2,1} + e_4\xi_{2,2}, \\ f(\xi) &= \check{f} + f_1\xi_{1,1} + f_2\xi_{1,2} + f_3\xi_{2,1} + f_4\xi_{2,2}. \end{aligned}$$

A simple calculation shows that if

$$\begin{aligned} h(\xi) &= a(\xi)\cdot b(\xi), \text{ and} \\ h_1(\xi) &= c(\xi)\cdot e(\xi) - d(\xi)\cdot f(\xi), \\ h_2(\xi) &= c(\xi)\cdot f(\xi) + e(\xi)\cdot d(\xi), \end{aligned}$$

then the relation $h(\xi) = h_1(\xi) + ih_2(\xi)$ holds, where $\xi_j = \left[\xi_{j,1}, \xi_{j,2}\right]$, $j = 1, 2$.

Let us comment on what the main ingredients of the described algorithm are. First, the information about the partial with respect to the initial condition derivative is obtained by introducing the first order jets in place of ordinary scalar values. Second, the first order jets are passed through the FFT algorithm in order to minimize the number of operations needed to calculate the convolutions. Third, in order to allow a fast calculation of the automatic differentiation convolutions the L_2 coefficients of the normalized derivatives are calculated and stored at each step.

Notation Let Φ_q^{jet} denote the extended variant of Φ_q—the q-th order Taylor method, whose coefficients are $\left\{\mathbf{J}_{\varphi^{[0]}}(\boldsymbol{\xi}), \ldots, \mathbf{J}_{\varphi^{[q-1]}}(\boldsymbol{\xi})\right\}$. In order to facilitate the explanation of the following algorithm we are going to use the following auxiliary symbols: $a^{[j]}(\boldsymbol{\xi}) := \mathbf{J}_{\varphi^{[j]}}(\boldsymbol{\xi})$, $N^{[j]}(\boldsymbol{\xi})$ denotes the first order jet for j-th normalized derivative of the non-linear part of (16), $N^{[j]}(\boldsymbol{\xi}) := \mathbf{J}_{(\mathbf{N}\circ\mathbf{a})^{[j]}}(\boldsymbol{\xi})$. $F^{[j]}(\boldsymbol{\xi})$ denotes the first order jet for the j-th normalized derivative of the right-hand side of (16) $F^{[j]}(\boldsymbol{\xi}) := \mathbf{J}_{(\mathbf{F}\circ\mathbf{a})^{[j]}}(\boldsymbol{\xi})$, where $F \circ a$ is the composition of F with a as a function of t. We indicate the corresponding L_2 coefficients with hats $\hat{a}^{[j]}(\boldsymbol{\xi})$, $\hat{N}^{[j]}(\boldsymbol{\xi})$ and $\hat{F}^{[j]}(\boldsymbol{\xi})$.

Algorithm 1 *Outline of an algorithm for efficient and rigorous calculation of one step of Φ_q^{jet}, dedicated for any system of the type (16).*

Input

- *the Galerkin projection dimension $N > 0$,*
- *number of points used by FFT, such that the aliasing problem is avoided (see Sect. 3.1) $M \geq 2N + 1$,*
- *an interval set of initial conditions $[x_0] \subset \mathbb{C}^{2N+1}$,*
- *time step $h > 0$,*
- *order of the Taylor method $q > 0$.*

Output

- *bounds for the normalized derivatives $\left\{\left(a_{-N}^{[j]}(\boldsymbol{\xi}), \ldots, a_N^{[j]}(\boldsymbol{\xi})\right)\right\}_{j=1}^{q}$, the coefficients of $\Phi_q^{jet}(h, [x_0])$,*
- *vector of first order jets $\Phi_q^{jet}(h, [x_0])$.*

begin

1. *Initialize calculations for Φ_q^{jet}. The base part of the jets $\left(a_{-N}^{[0]}(\boldsymbol{\xi}), \ldots, a_N^{[0]}(\boldsymbol{\xi})\right)$ is set to be equal to the provided initial condition $[x_0]$. The variational part of the jets $\left(a_{-N}^{[0]}(\boldsymbol{\xi}), \ldots, a_N^{[0]}(\boldsymbol{\xi})\right)$ is set to be the identity, i.e.*

$$\frac{\partial \operatorname{Re}\left(\varphi_k^{[0]}\right)}{\partial \operatorname{Re}(a_l)} := \begin{cases} 1 \ , \ for\ k = l \\ 0 \ otherwise. \end{cases}, \quad \frac{\partial \operatorname{Im}\left(\varphi_k^{[0]}\right)}{\partial \operatorname{Im}(a_l)} := \begin{cases} 1 & , \ for\ k = l, \\ -1 & , \ for\ k = -l, \\ 0 & otherwise. \end{cases},$$

$$\frac{\partial \operatorname{Re}\left(\varphi_k^{[0]}\right)}{\partial \operatorname{Im}(a_l)}, \frac{\partial \operatorname{Im}\left(\varphi_k^{[0]}\right)}{\partial \operatorname{Re}(a_l)} := 0.$$

2. *$l_2 \to L_2$ transform $\left(a_{-N}^{[0]}(\boldsymbol{\xi}), \ldots, a_N^{[0]}(\boldsymbol{\xi})\right)$ to obtain $\left(\hat{a}_0^{[0]}(\boldsymbol{\xi}), \ldots, \hat{a}_M^{[0]}(\boldsymbol{\xi})\right)$*
   for *$j = 0, \ldots, q-1$*
   (a) *calculate $\left(\hat{N}_0^{[j]}(\boldsymbol{\xi}), \ldots, \hat{N}_M^{[j]}(\boldsymbol{\xi})\right)$ using the L_2 coefficients calculated in the previous steps $\left\{\left(\hat{a}_0^{[0]}(\boldsymbol{\xi}), \ldots, \hat{a}_M^{[0]}(\boldsymbol{\xi})\right), \ldots, \left(\hat{a}_0^{[j]}(\boldsymbol{\xi}), \ldots, \hat{a}_M^{[j]}(\boldsymbol{\xi})\right)\right\}$,*
   (b) *calculate $\left(N_{-N}^{[j]}(\boldsymbol{\xi}), \ldots, N_N^{[j]}(\boldsymbol{\xi})\right)$ by $L_2 \to l_2$ transforming $\left(\hat{N}_0^{[j]}(\boldsymbol{\xi}), \ldots, \hat{N}_M^{[j]}(\boldsymbol{\xi})\right)$,*
   (c) *add the linear contribution*
   $$\left(F_{-N}^{[j]}(\boldsymbol{\xi}), \ldots, F_N^{[j]}(\boldsymbol{\xi})\right) := \left(N_{-N}^{[j]}(\boldsymbol{\xi}), \ldots, N_N^{[j]}(\boldsymbol{\xi})\right) + \left(\lambda_{-N} a_{-N}^{[j]}(\boldsymbol{\xi}), \ldots, \lambda_N a_N^{[j]}(\boldsymbol{\xi})\right),$$

(d) *set*
$\left(a_{-N}^{[j+1]}(\boldsymbol{\xi}),\ldots,a_{N}^{[j+1]}(\boldsymbol{\xi})\right) := \frac{1}{j+1}\left(F_{-N}^{[j]}(\boldsymbol{\xi}),\ldots,F_{N}^{[j]}(\boldsymbol{\xi})\right)$,

(e) *calculate and store* $\left(\hat{a}_{0}^{[j+1]}(\boldsymbol{\xi}),\ldots,\hat{a}_{M}^{[j+1]}(\boldsymbol{\xi})\right)$ *by* $l_2 \to L_2$ *transforming* $\left(a_{-N}^{[j+1]}(\boldsymbol{\xi}),\right.$
$\left.\ldots,a_{N}^{[j+1]}(\boldsymbol{\xi})\right)$,

end for

3. *calculate the value of* $\Phi_q^{jet}(h,[x_0])$ *using the coefficients from the previous step and Horner's rule.*

end

Remark 1 Let us relate the output results of Algorithm 1 to the mean value form formulation (10). The matrix composed of the variational part of the vector of first order jets $\Phi_q^{jet}(h,[x_0])$ is the factor $\partial\Phi_q/\partial x\,(h,[x])$ in (10). The other terms in (10), i.e. $\Phi_q(h,\mathrm{mid}\,([x]))$, $R_{\Phi_q}([W])$ are obtained by two separate Taylor methods, as different initial conditions has to be used. The vectors $\Phi_q(h,\mathrm{mid}\,([x]))$, and $R_{\Phi_q}([W])$ are evaluated using scalar values, not first order jets, and when N is large the cost of calculating them is negligible compared to the cost of calculating $\Phi_q^{jet}(h,[x_0])$.

Now we would like to comment on Step (a) of the "for" loop in Algorithm 1, because that is where savings are introduced by the FFT algorithm.

The improvement of introducing the FFT algorithm into the standard procedure is as follows—at the given order j instead of calculating the double convolutions

$$\sum_{k_1+k_2+\cdots+k_n=k}\ \sum_{j_1+j_2+\cdots+j_n=j} a_{k_1}^{[j_1]}(\boldsymbol{\xi})\,a_{k_2}^{[j_2]}(\boldsymbol{\xi})\cdots a_{k_n}^{[j_n]}(\boldsymbol{\xi}),\quad k=-N,\ldots,N \tag{18}$$

directly, the following operations, comprising steps 2a, 2b and 2e of Algorithm 1 or equivalently comprising steps (a), (b) and (e) of Algorithm 1 are used, compare with the outline presented in Sect. 2.1:

– one $l_2 \to L_2$ transform, and one $L_2 \to l_2$ transform,
– calculation of

$$\sum_{j_1+j_2+\cdots+j_n=j} \hat{a}_{k}^{[j_1]}(\boldsymbol{\xi})\,\hat{a}_{k}^{[j_2]}(\boldsymbol{\xi})\ldots\hat{a}_{k}^{[j_n]}(\boldsymbol{\xi}),\qquad k=0,\ldots,M. \tag{19}$$

Observe that in the FFT variant the loop over k is not present at all.

We discuss in [8] several optimizations that can be introduced into this algorithm.

5 Rigorous Numerics Tests

Below we present a report from various tests we performed with the developed methods and algorithms. We emphasize that by tests we mean rigorous numerics tests. The purpose of the developed methods and algorithms are the computer assisted proofs for dPDEs. In the tests either a finite truncation (a *Galerkin projection*) or the full infinite system (giving a *differential inclusion*) were considered. In order to integrate in time the full infinite dimensional system of ODEs (2) we split it into two parts

$$\begin{cases} \frac{dx}{dt} = P_N F(x+y), \\ \frac{dy}{dt} = Q_N F(x+y), \end{cases} \tag{20}$$

where $F(x+y)$ is the right-hand side of (2), $X_N = \mathbb{C}^{2N+1}$, $Y_N \subset \mathbb{C}^\infty$, $x \in X_N$, $y \in Y_N$, P_N is the projection onto X_N, $Q_N = Id - P_N$. We assume the embedding $X_N \ni \{x_{-N}, \ldots, x_0, \ldots, x_N\} \equiv \{0, \ldots, 0, x_{-N}, \ldots, x_0, \ldots, x_N, 0, \ldots, 0\} \in \mathbb{C}^\infty$, and use the same symbol to denote both of the elements of the finite dimensional space and the infinite dimensional space. We bound the solutions of (2) by considering the following *differential inclusion*

$$\frac{dx}{dt}(t) \in P_N(F(x(t))) + \delta, \tag{21}$$

where $\delta \subset X_N$ describes the influence of y onto $P_N F(x+y)$. The Lohner-type algorithm for rigorous integration in time of differential inclusions was originally provided in [19]. In this algorithm the solutions of a Cauchy problem associated with (21) are bounded, and then the influence of the perturbation δ is a-posteriori added. In order to calculate δ the FFT algorithm may also be used.

We specified what system was being integrated for each test separately by stating a "Galerkin projection" or a "differential inclusion" respectively. We used the implementation of the part of the Lohner algorithm regarding the Lohner representation of the sets from the CAPD library [4].

In this section we present results from all the tests performed in the form of tables. Whenever a diameter of a finite or an infinite dimensional set is provided in a table, the meaning of this number is in fact the maximum over coordinates of all diameters. The labels *"Direct approach" "FFT approach"* and *"FadBad++"* appearing in the tables indicate different implementations of the Taylor method that were used. "Direct approach" indicates that the normalized derivatives were obtained directly (18), whereas "FFT approach" indicates that the normalized derivatives were obtained by the FFT approach presented previously. To remove the aliasing error technique the "II technique based on phase shifts" was used, unless otherwise stated. "FadBad++" indicates that the normalized derivatives were obtained by using the FadBad++ software library for automatic differentiation [2], without using the FFT at any point. The vector field, which was input to the FadBad++ library was optimized, by using Observation 1 and all possible symmetries. The part of the Lohner algorithm regarding the Lohner representation of the sets was the same for both. For a thorough description of the Lohner algorithm and the issues related to the wrapping effect and representation of sets refer to [33]. The meaning of the symbol M depends on the context; in [35] and [7] it is used to denote the dimension of the so-called finite tail, when the full infinite dimensional system is considered, whereas in this paper it is used to denote the number of points of the discrete grid used by the FFT. To avoid ambiguities in this section, we denote the number of points of the discrete grid used by the FFT by M_{FFT}.

5.1 Burgers Equation Fixed Point Test (Galerkin Projection)

When periodic boundary conditions are used and the forcing function is not present, the Burgers equation exhibits a globally attracting fixed point at zero. This is proved by the energy exponential decay, see [7].

With this test we address the question of what are, for the overall Lohner algorithm, the consequences of considerable overestimates produced by the FFT approach (see the results from Sect. 3.4). This question is difficult to answer without performing actual calculations due to the sophisticated nature of the Lohner type rigorous integrator.

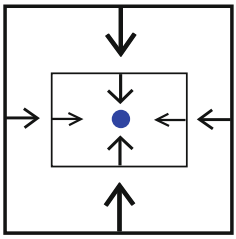

Fig. 1 The setting for the Burgers equation

The setting is

$$u_t + u \cdot u_x - \nu u_{xx} = 0, \tag{22a}$$
$$u(x,0) = u_0(x), \quad x \in \mathbb{R}, \tag{22b}$$
$$u(x,t) = u(x + 2k\pi, t), \quad x \in \mathbb{R},\ t \in [0, \mathcal{T}),\ k \in \mathbb{Z}. \tag{22c}$$

The uniform zero-centred box (with the fixed point at the centre) was rigorously integrated forward in time on the fixed time interval with a fixed time step. The diameter of the initial box was the parameter of the tests (Fig. 1).

In Table 4 we present a report from the performed tests. The diameter of the result, which was, in fact, the diameter of the set at the time 10 (10, 000 constant time steps of length 0.001 were executed), was measured. Due to the fact that the fixed point is attracting, the initial box is expected to decrease in diameter. We were in particular interested in the difference between the quality of the FFT approach result, and the direct approach result. The results presented in Sect. 3.4 predict much better quality of the direct approach result.

The parameters that were used for the tests were $\nu = 0.1$, order of the Taylor method was 6, constant time-step $h = 0.001$, $N = 19$, $M_{FFT} = 40$.

Total running time	
FadBad++	FFT approach
1849 sec	424 sec

(23)

See (23) for the comparison of the program running time.

5.2 Kuramoto-Sivashinsky Equation Attracting Periodic Orbit Test

The setting is

$$u_t = -\nu u_{xxxx} - u_{xx} + (u^2)_x, \tag{24a}$$
$$u(x,0) = u_0(x), \quad x \in \mathbb{R}, \tag{24b}$$
$$u(x,t) = -u(-x,t), \quad u(x,t) = u(x + 2k\pi, t), \quad x \in \mathbb{R},\ t \in [0, \mathcal{T}),\ k \in \mathbb{Z}. \tag{24c}$$

We picked one of the periodic orbits proved to exist in [35] (the attracting periodic orbit for $\nu = 0.032$). We took a box around an initial condition from [35] for this periodic orbit, and then the box was rigorously propagated forward in time using two implementations of the Lohner algorithm. In the tests the equations were integrated until the time of the full revolution of orbit was reached, which is approximately 0.4091. The results of these tests are contained in Tables 5 and 6.

I test—efficiency (Galerkin projection) All of the techniques presented previously in this part of the article can be replaced by applying some of the available software libraries for

Table 4 Fixed point for Burgers equation test results

#	Initial condition diam.	Set at the end diam.	
		direct approach	FFT approach
0	1e − 08	9.04837e − 09	9.04838e − 09
1	1e − 07	9.04838e − 08	9.0484e − 08
2	1e − 06	9.0484e − 07	9.04867e − 07
3	1e − 05	9.04862e − 06	9.05131e − 06
4	0.0001	9.05088e − 05	9.0789e − 05
5	0.001	0.000907367	0.000958336
6	0.01	0.00932532	0.00932849[a]

[a]In this test a blow-up of the set was experienced for the FFT approach. By a blow-up we mean that the rigorous bounds escaped the range of the representable numbers. Such bounds are then represented in the software by the unbounded interval ($[-\infty, \infty]$). As soon as the unbounded interval appears in the calculation process, further calculations are meaningless. We found a simple method circumventing this problem at a little additional computational cost. To avoid the blow-up problem, the first order normalized derivative must be calculated by using the direct approach, nonetheless the higher order derivatives can be obtained by using the FFT approach. The cost of calculating the first order normalized derivative is negligible for the overall cost of calculating the normalized derivatives up to a reasonable fixed order

Table 5 Total running time of programs propagating a box along an orbit for the KS equation (an attracting periodic orbit for $\nu = 0.032$)

	Parameters used				Total running time	
#	Galerkin proj. dim.	M_{FFT}	Time step	Taylor m. ord.	FadBad++	FFT approach
1	23	48	10^{-4}	5	139 s	107 s
2	23	48	10^{-4}	8	297 s	175 s
3	23	48	10^{-4}	10	438 s	229 s
4	28	60	5^{-5}	5	496 s	343 s
5	34	72	2^{-5}	5	2147 s	1295 s

The machine used was Linux 32-bit Intel Core i5-2430M CPU @ 2.40 GHz x 4

automatic differentiation, for instance Fad-Bad++ [2]. The purpose of this test was to check if the presented approach will eventually be useful in calculating a rigorous bounds of the Poincaré map, which are needed e.g. to verify the Brouwer fixed point theorem assumptions. With these tests we would like to provide an argument that we did not perform redundant work developing this approach. The Lohner algorithm implementation based upon the FFT approach allowed to obtain bounds of the same order, and at the same time this bounds were obtained faster.

With this test we compared the efficiency of the Lohner algorithm implementation based upon the presented FFT approach with the implementation based on the direct approach implemented using the external software library for automatic differentiation FadBad++ [2] (the approach used to perform the computer assisted proof presented in [7]).

A Galerkin projection of the infinite dimensional system was considered in these tests in order to isolate this part of the overall algorithm to which the developed techniques contribute.

The initial condition was taken from [35], the diameter of the initial condition was $8e-05$ in each direction. Parameters used for this test were as follows $\nu = 0.032$, order of the

Table 6 Diameter (max over coordinates) of boxes obtained by programs propagating a box along orbit for KS equation (an attracting periodic orbit for $\nu = 0.032$)

#	Initial condition diam.	Set at the end diam.	
		Direct approach	FFT approach
0	1e − 08	7.7879e − 05	0.00012947
1	1e − 07	9.55487e − 05	0.000134767
2	1e − 06	0.000174207	0.00025008
3	1e − 05	0.000231317	0.00309667
4	0.0001	0.00134516	0.00221152[a]

The full infinite dimensional system was considered (differential inclusion)
[a]In this test a blow-up of the set was experienced for the FFT approach. By a blow-up we mean that the rigorous bounds escaped the range of the representable numbers. Such bounds are then represented in the software by the unbounded interval ($[-\infty, \infty]$). As soon as the unbounded interval appears in the calculation process, further calculations are meaningless. We found a simple method circumventing this problem at a little additional computational cost. To avoid the blow-up problem, the first order normalized derivative must be calculated by using the direct approach, nonetheless the higher order derivatives can be obtained by using the FFT approach. The cost of calculating the first order normalized derivative is negligible for the overall cost of calculating the normalized derivatives up to a reasonable fixed order

Taylor method was 5, constant time-step $h = 0.00015$, $N = 23$ (all the same as in [35]), $M_{FFT} = 48$.

Observe that the time step had to be adjusted for each test, because of the stiffness problem exhibited by the Kuramoto-Sivashinsky equation. The results of this test are contained in Table 5.

II test—overestimates (differential inclusion) The diameter of the propagated set was measured after the full revolution along the orbit. The full rigorous integrator (the Lohner algorithm for differential inclusions) of the infinite dimensional system was used in this test.

Parameters used for this test were as follows $\nu = 0.032$, order of the Taylor method was 5, constant time-step $h = 0.00015$, $N = 23$, $M = 92$ (the dimension of the finite tail, for the exact meaning of this number refer to [35] or [7]), $M_{FFT} = 48$. The results of this test are contained in Table 6.

5.3 Kuramoto-Sivashinsky Equation Unstable Periodic Orbit Test (Differential Inclusion)

The same setting as in Sect. 5.2.

We picked one of the unstable periodic orbits proved to exist in [35] (the unstable periodic orbit for $\nu = 0.02991$). This orbit is on the apparently chaotic attractor. We took a box around an initial condition from [35] for this periodic orbit, and then the box was rigorously propagated forward in time using two implementations of the Lohner algorithm. In the tests the equations were integrated until the time of the full revolution of the orbit was reached, which is approximately 0.4490232. The results of these tests are contained in Table 7.

The diameter of the propagated set was measured after the full revolution along the orbit. The full rigorous integrator (the Lohner algorithm for differential inclusions) of the infinite dimensional system was used in this test.

Parameters used for this test were as follows $\nu = 0.02991$, order of the Taylor method was 5, constant time-step $h = 0.00015$, $N = 25$, $M = 75$ (the dimension of the finite tail, for the exact meaning of this number refer to [35] or [7]), $M_{FFT} = 64$.

Table 7 Diameter (max over coordinates) of boxes obtained by programs propagating a box along orbit for KS equation (an unstable periodic orbit for $\nu = 0.02991$)

#	Initial condition diam.	Set at the end diam.	
		Direct approach	FFT approach
0	1e − 08	0.000172909	0.000218735
1	1e − 07	0.000215746	0.000247145
2	1e − 06	0.000352979	0.000558827
3	1e − 05	0.00051132	0.00801811
4	0.0001	0.00406078	0.0049136[a]

The orbit is on the apparently chaotic attractor. The full infinite dimensional system was considered (differential inclusion)

[a]In this test a blow-up of the set was experienced for the FFT approach. By a blow-up we mean that the rigorous bounds escaped the range of the representable numbers. Such bounds are then represented in the software by the unbounded interval $([-\infty, \infty])$. As soon as the unbounded interval appears in the calculation process, further calculations are meaningless. We found a simple method circumventing this problem at a little additional computational cost. To avoid the blow-up problem, the first order normalized derivative must be calculated by using the direct approach, nonetheless the higher order derivatives can be obtained by using the FFT approach. The cost of calculating the first order normalized derivative is negligible for the overall cost of calculating the normalized derivatives up to a reasonable fixed order

5.4 Swift-Hohenberg Equation a Connection Between Fixed Points Test (Differential Inclusion)

The setting is

$$u_t = \left(\nu - \left(1 + \frac{\partial^2}{\partial x^2}\right)^2\right) u - u^3, \tag{25a}$$

$$u(x, 0) = u_0(x), \quad x \in \mathbb{R}, \tag{25b}$$

$$u(x, t) = u(-x, t), \quad u(x, t) = u(x + 2k\pi, t), \quad x \in \mathbb{R},\ t \in [0, \mathcal{T}),\ k \in \mathbb{Z}. \tag{25c}$$

Looking at the eigenvalues of the linear part of (25)

$$\lambda_k = \nu - (1 - k^2)^2,$$

it is immediately verified that zero is an unstable fixed point for (25a), for a sufficiently large ν. Apparently, there are also other fixed points, which are attracting.

Overestimates (differential inclusion) As the initial condition, we took a small interval set of functions close to zero, i.e.

$$u_0 = 2 \cdot \left(10^{-10} + [-10^{-11}, 10^{-11}]\right) + 2 \cdot \left(10^{-10} + [-10^{-11}, 10^{-11}]\right) \cos x.$$

This set of functions had been rigorously integrated until it was caught in the basin of attraction of an attracting fixed point for (25a). The process of integration of the set was stopped as soon as the set was attracted by and centred at an attracting fixed point location (Fig. 2).

The order of the Taylor method used for the tests was 5. The results of this test are contained in Table 8.

Numerical data from all the tests presented in this section is available on-line [28].

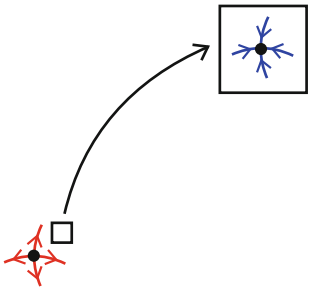

Fig. 2 The setting for the Swift-Hohenberg equation

Table 8 Diameter (max over coordinates) of boxes obtained by programs propagating a box along connection between fixed points for SH equation

#	Parameters used					Set at end diam.	
	ν	Number of steps	Time step	N	The finite tail dim.	Direct approach	FFT approach
0	25	15000	0.0001	11	33	2.31318e − 05	4.62783e − 05[a]
1	50	10000	0.0001	11	33	1.28577e − 06	3.19921e − 05[a]
2(a)	65	6000	0.0001	12	50	0.000153006	0.00365307[b]
2(b)	65	6000	0.0001	12	50	0.000153006	0.00339236[c]

[a]In this test a blow-up of the set was experienced for the FFT approach. By a blow-up we mean that the rigorous bounds escaped the range of the representable numbers. Such bounds are then represented in the software by the unbounded interval ($[-\infty, \infty]$). As soon as the unbounded interval appears in the calculation process, further calculations are meaningless. We found a simple method circumventing this problem at a little additional computational cost. To avoid the blow-up problem, the first order normalized derivative must be calculated by using the direct approach, nonetheless the higher order derivatives can be obtained by using the FFT approach. The cost of calculating the first order normalized derivative is negligible for the overall cost of calculating the normalized derivatives up to a reasonable fixed order

[b]This is the same test as in 2(b). In this test a blow-up was experienced. It was possible to eliminate it by using the technique described in foot note a like in other cases, but with proper choice of M—the number of discrete points used by FFT (in this case for $M = 32$ the blow-up was eliminated, whereas for $M = 30$ the blow-up was still experienced)

[c]This is the same test as in 2(a). In this test a blow-up was experienced. It was eliminated when further the padding technique was used instead of the phase-shift aliasing error elimination technique regardless the choice of M—the number of discrete points used by FFT

6 Complexity

In this section we analyse the complexity of the direct approach and the FFT approach, which were compared in numerical tests presented in Sect. 5.

First, we compare the asymptotic complexity of one step (generating the coefficients and evaluating the polynomial) of the extended Taylor method of the order q (Φ_q^{jet}). Coefficients of Φ_q^{jet} are the first order jets of a vector valued function. Second, we compare the actual number of elementary operations required to perform one step in both of the approaches. We calculated the actual number of operations, motivated by the fact that asymptotic complexity provides only a rough information about the magnitude of inputs that can be handled by using the current computing power, and we have been interested explicitly how much faster we can calculate.

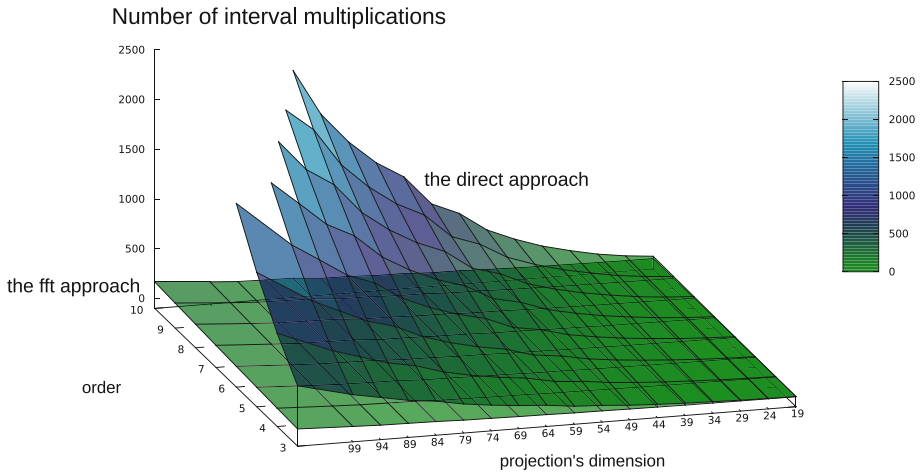

Fig. 3 The number of interval multiplications (divided by 10^6) required to perform one step of Φ_q^{jet} in the Burgers equation setting (22) with periodic boundary conditions

We use the well known fact that the FFT algorithm's asymptotic complexity is $O(n \log n)$, where n is the size of the input. The asymptotic number of operations required to perform one step of Φ_q^{jet} using the FFT approach is

$$\overbrace{c_1 q N M \log M}^{\text{FFT's on jets}} + \overbrace{c_2 q(q+1) M N}^{\text{multiplications of the } L_2 \text{ coefficients (jets) in FFT's}}, \tag{26}$$

whereas the complexity of one step of Φ_q^{jet} using the direct approach is

$$\overbrace{c_3 q(q+1) N^3}^{\text{"double" convolution on jets}}, \tag{27}$$

where $c_1, c_2, c_3 > 0$ are constants, and M is a number depending linearly on N, such that $M \geq 2N + 1$, see Sect. 3.1. The term $NM \log M$ in (26) in place of $M \log M$ seems strange at the first sight, but it is due to the fact that the input and the output for all the FFT's are vectors of first order jets. The number of elementary operations required to perform an arithmetic operation on first order jet depends linearly on N, as one loop is required—compare Definition 6. For the meaning of "the FFT approach" and "the direct approach" refer to Sect. 5.

To confirm (26) and (27) empirically we counted the exact number of interval multiplications and interval additions, which are performed by the computer program during one step of Φ_q^{jet} in the Burgers equation setting (22). The results are presented in Fig. 3. Observe that the calculations are performed within the interval arithmetic framework. An interval arithmetic operation is considerably more costly than the corresponding ordinary floating-point arithmetic operation, e.g. to perform an interval multiplication usually four floating-point multiplications and two rounding mode switchings are required. Overall tests indicate that the execution time of an interval arithmetic operation is approximately of the order ten times the time of the corresponding floating-point arithmetic operation.

We also counted the number of interval additions and multiplications for the other two settings, i.e. (24) and (25). But the corresponding figures are similar in spirit to Fig. 3, so we

Table 9 Exact number of elementary operations on complex numbers required to calculate the j-th coefficient of Φ_q^{jet}, which is one step of the "for" loop in Algorithm 1

Approach	Additions	Multiplications
FFT	$\left[2Mp + M\left(\lceil \frac{j}{2} \rceil - 1\right)\right](4N+3)$ $+M\lceil \frac{j}{2} \rceil (4N+2)$	$M\lceil \frac{j}{2} \rceil (8N+5)+$ $(Mp + M + 2N + 1)(4N+3)$
direct	$\left[\left(\frac{3}{2}N^2 - N + 2\left\lfloor \frac{N}{2} \right\rfloor\right)(4N+3)+\right.$ $\left.\left(\frac{3}{2}N^2 - N + 2\left\lfloor \frac{N}{2} \right\rfloor\right)(4N+2)\right] j - 2N(4N+3)$	$\left[\left(\frac{3}{2}N^2 - N + 2\left\lfloor \frac{N}{2} \right\rfloor\right)(8N+5)\right] j+$ $2(2N+1)(4N+3)$

The calculations were performed in the Burgers equation setting (22), p is a number such that $M = 2^p$

do not present them here and refer the interested reader to the corresponding numerical data [28].

In order to get an idea about the magnitude of constants c_1, c_2, c_3 in (26) and (27) we provided in [8] the exact number of elementary operations on complex numbers required to calculate the j-th coefficient of Φ_q^{jet}. The calculations were performed in the Burgers equation setting (22), and assuming that $M = 2^p$. We present the results in Table 9.

7 Conclusion

We would like to present some concluding remarks from the results presented above.

The diameters of the sets obtained in the performed tests (Tables 4, 6, and 8) indicate that, surprisingly, the Lohner algorithm seems to be immune to the considerable overestimates produced by the FFT approach. Consequently, we are convinced that the Lohner algorithm based on the developed here FFT approach can be successfully applied to the problem of rigorous integration of higher dimensional PDEs, including the Navier-Stokes equations. And the presented algorithm perfectly suits performing proofs of an existence of solution on a fixed time interval, for instance when one needs to establish existence of a Poincaré map only.

The different test cases have demonstrated that the software based on the developed techniques has improved the efficiency of the rigorous integration task, without severe quality drop.

All the test cases were successfully finished when the new approach was used (no interval blow-ups were experienced), and there was not any considerable quality drop (measured by diameter of the propagated interval sets). We consider this a significant achievement in the context of eventual computer assisted proofs for higher dimensional PDEs. The task of rigorous integration of a higher dimensional PDEs like the Navier-Stokes equation, which is our goal, will require a large amount of computing time. Any reduction of the time spent on computations is of great importance here, because this could allow some phenomena to be actually proven.

Different equations and different boundary conditions were used in each of the presented tests (the Burgers equation with a pure periodic boundary conditions, the KS equation with a periodic-odd boundary conditions and the SH equation with a periodic-even boundary conditions) in order to demonstrate that the developed software is generic and robust. The software can be applied for many problems involving dPDEs.

The goal of achieving both generic and efficient software for rigorous integration of dPDEs has been achieved to some extent. It is a difficult task in general. We have been able to achieve

this for periodic boundary conditions in 1D. Still, such software had not existed, and all the results published so far (e.g. [1,35]) have focused on a particular case of the Kuramoto-Sivashinsky equation, and the software used there was hard-coded for the KS equation with periodic-odd boundary conditions.

The tests discussed in previous section are in principle repeatable, files with data from the presented proofs and all the source code used are available on-line [28].

Acknowledgments Research has been supported by National Science Centre Grant DEC-2011/01/N/ST6/00995.

References

1. Arioli, G., Koch, H.: Integration of dissipative partial differential equations: a case study. SIAM J. Appl. Dyn. Syst. **9**, 1119–1133 (2010)
2. Bendtsen, C., Stauning, O.: FADBAD, a Flexible C++ Package for Automatic Differentiation. Technical University of Denmark, Department of Mathematical Modelling, Copenhagen (1996)
3. Breuer, B., McKenna, P.J., Plum, M.: Multiple solutions for a semilinear boundary value problem: a computational multiplicity proof. J. Differ. Equ. **195**, 243–269 (2003)
4. CAPD—computer assisted proofs in dynamics, a package for rigorous numeric, http://capd.ii.uj.edu.pl
5. Canuto, C., Hussaini, M.Y., Quarteroni, A., Zang, T.A.: Spectral Methods. Fundamentals in Single Domains. Springer, Berlin (2006)
6. Cooley, J.W., Tukey, J.W.: An algorithm for the machine calculation of complex fourier series. Math. Comp. **19**, 297–301 (1965)
7. Cyranka, J.: Existence of Globally Attracting Fixed Points of Viscous Burgers Equation with Constant Forcing. A Computer Assisted Proof, preprint, 2011, available on-line http://www.ii.uj.edu.pl/~cyranka
8. Cyranka, J.: Efficient Algorithms for Rigorous Integration Forward in Time of dPDEs. Existence of Globally Attracting Fixed Points of Viscous Burgers Equation with Constant Forcing, a Computer Assisted Proof, PhD dissertation, Jagiellonian University, Cracow, 2013, available on-line http://www.ii.uj.edu.pl/~cyranka
9. Day, S., Hiraoka, Y., Mischaikow, K., Ogawa, T.: Rigorous numerics for global dynamics: a study of Swift-Hohenberg equation. SIAM J. Appl. Dyn. Syst. **4**, 1–31 (2005)
10. Day, S., Lessard, J.P., Mischaikow, K.: Validated continuation for equilibria of PDEs. SIAM J. Numer. Anal. **45**, 1398–1424 (2007)
11. Gameiro, M., Lessard, J.P., Mischaikow, K.: Validated continuation over large parameter ranges for equilibria of PDEs. Math. Comput. Simul. **79**, 1368–1382 (2008)
12. Gameiro, M., Lessard, J.P.: Analytic estimates and rigorous continuation for equilibria of higher-dimensional PDEs. J. Differ. Equ. **249**, 2237–2268 (2010)
13. Gameiro, M., Lessard, J.P.: Rigorous computation of smooth branches of equilibria for the three dimensional Cahn-Hilliard equation. Numer. Math. **117**, 753–778 (2011)
14. Griewank, A.: Evaluating Derivatives. Principles and Techniques of Algorithmic Differentiation, Frontiers in Applied Mathematics, vol. 19. SIAM, Philadelphia (2000)
15. Heywood, J.G., Nagata, W., Xie, W.: A numerically based existence theorem for the Navier-Stokes equations. J. Math. Fluid. Mech. **1**, 5–23 (1999)
16. Hiraoka, Y., Ogawa, T.: Rigorous numerics for localized patterns to the quintic Swift-Hohenberg equation, Japan. J. Ind. Appl. Math. **22**, 57–75 (2005)
17. Hiraoka, Y., Ogawa, T.: An efficient estimate based on FFT in topological verification method. J. Comput. Appl. Math. **199**, 238–244 (2007)
18. Jorba, Á., Zou, M.: A software package for the numerical integration of ODEs by means of high-order Taylor methods. Exp. Math. **14**, 99–117 (2005)
19. Kapela, T., Zgliczyński, P.: A Lohner-type algorithm for control systems and ordinary differential inclusions. Discret. Cont. Dyn. Sys. B **11**, 365–385 (2009)
20. Kim, M., Nakao, M.T., Watanabe, Y., Nishida, T.: A numerical verification method of bifurcating solutions for 3-dimensional Rayleigh-Bénard problems. Numer. Math. **111**, 389–406 (2009)

21. Lohner, R.J.: Computation of guaranteed enclosures for the solutions of ordinary initial and boundary value problems. In: Cash, J.R., Gladwell, I. (eds.) Computational Ordinary Differential Equations. Clarendon Press, Oxford (1992)
22. Maier-Paape, S., Miller, U., Mischaikow, K., Wanner, T.: Rigorous numerics for the Cahn-Hilliard equation on the unit square. Rev. Mat. Complut. **21**, 351–426 (2008)
23. Moore, R.E.: Interval Analysis. Prentice-Hall, New Jersey (1966)
24. Moore, R.E.: Methods and Applications of Interval Analysis. SIAM, Philadelphia (1979)
25. Nakao, M.T.: Numerical verification methods for solutions of ordinary and partial differential equations. Numer. Funct. Anal. Optim. **22**, 321–356 (2001)
26. Orszag, S.A., Patterson, G.S.: Spectral calculations of isotropic turbulence: efficient removal of aliasing interactions. Phys. Fluids **14**, 2538–2541 (1971)
27. Simó, C.: Taylor Method for the Integration of ODE. Lectures given at the LTI07 Advanced School on Long Time Integrations, available online http://www.maia.ub.es/dsg/2007/
28. Software package and the data from tests, http://www.ii.uj.edu.pl/~cyranka/FFT
29. Temperton, C.: Self-sorting mixed-radix fast fourier transforms. J. Comput. Phys. **52**, 1–23 (1983)
30. Temperton, C.: Fast mixed-radix real fourier transforms. J. Comput. Phys. **52**, 340–350 (1983)
31. Warmus, M.: Calculus of Approximations. Bull. de l'Académie Polonaise des Sciences Cl. III **4**, 253–259 (1956)
32. Zgliczyński, P.: Attracting fixed points for the Kuramoto-Sivashinsky equation—a computer assisted proof. SIAM J. Appl. Dyn. Syst. **1**, 215–288 (2002)
33. Zgliczyński, P.: C^1-Lohner algorithm. Found. Comput. Math. **2**, 429–465 (2002)
34. Zgliczyński, P.: Rigorous numerics for dissipative partial differential equations II. Periodic orbit for the Kuramoto-Sivashinsky PDE—a computer assisted proof. Found. Comput. Math. **4**, 157–185 (2004)
35. Zgliczyński, P.: Rigorous numerics for dissipative PDEs III. An effective algorithm for rigorous integration of dissipative PDEs. Topol. Methods Nonlinear Anal. **36**, 197–262 (2010)
36. Zgliczyński, P., Mischaikow, K.: Rigorous numerics for partial differential equations: the Kuramoto-Sivashinsky equation. Found. Comput. Math. **1**, 255–288 (2001)