Using Automatic Differentiation to compute periodic orbits of Delay Differential Equations*

Joan Gimeno and Àngel Jorba
Departament de Matemàtiques i Informàtica
Barcelona Graduate School of Mathematics (BGSMath),
Universitat de Barcelona
Gran Via de les Corts Catalanes 585
08007 Barcelona, Spain
E-mails: `joan@maia.ub.es`, `angel@maia.ub.es`

December 30th, 2019

Abstract

In this paper we focus on the computation of periodic solutions of Delay Differential Equations (DDEs) with constant delays. The method is based on defining a Poincaré section in a suitable functional space and looking for a fixed point of the flow in this section. This is done by applying a Newton method on a suitable discretization of the section. To avoid computing and storing large matrices we use a GMRES method to solve the linear system because in this case GMRES converges very fast due to the compactness of the flow of the DDE. The derivatives of the Poincaré map are obtained in a simple way, by applying Automatic Differentiation to the numerical integration. The stability of the periodic orbit is also obtained in a very efficient way by means of Arnoldi methods. The examples considered include temporal and spatial Poincaré sections.

*This work has been supported by the Spanish grants PGC2018-100699-B-I00 (MCIU/AEI/FEDER, UE) and the Catalan grant 2017 SGR 1374. The project leading to this application has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 734557.

Contents

1 Introduction

Delay Differential Equations (DDEs) theory have been studied in the mathematical literature by many authors, such as [Hal77, HMN91] and many others. However, many mathematical models with constant time delays or with state delays require complicated mathematical analysis to be treated only partially. As a result, such models require numerical methods to analyse the dynamical features; steady-state points, periodic orbits, bifurcation behaviours, etc. The development of these methods is, in general, difficult due to the properties of a delay dynamical system and it may be much more complicated than the non-delay case. Consequently, research on numerical techniques for DDEs has been mainly focused not only on time integration but also in the computation of invariant objects ([SEL$^+$15]).

The existence of periodic solutions is an important topic of interest in many applications of DDEs. There are many results and methods to study their existence, stability and dependence on parameters of certain classes of DDEs, [SW06]. However, many of these results and the corresponding methods are essentially theoretical and they cannot be easily applied to a general nonlinear system with several delays.

In this paper we focus on the computation of periodic orbits and their stability. The main idea is to compute the periodic orbit as a fixed point of a suitable Poincaré map by means of the Newton method. The main novelty is that the linear system that appears at each Newton iteration is solved by a GMRES method, which do not require the computation of the full Jacobian matrix, but only its action on given vectors (directional derivatives). These derivatives are computed by means of Automatic Differentiation (AD), a technique that is very suitable to propagate derivatives along a computer code, since it is very simple to implement [GC91, Gri00, Nau12].

We recall that the periodicity problem for DDEs is an infinite-dimensional problem because such a problem is defined in an infinite-dimensional space. Indeed, while only an initial point is required to characterise periodic solutions of an Ordinary Differential Equation (ODE), computing a delay periodic orbit requires an initial function to be found. Such computation will be done using a shooting approach [LELR97] discretizing an initial function on the delay interval and using a discrete version of the Poincaré operator whose eigenvalues will be approximations of the Floquet multipliers of the periodic solution of a likely large system. They will be clustered to zero and usually only a few of them have large modulus [Hal77].

1.1 Delay Differential Equations

A Delay Differential Equation (DDE) is a functional equation of the form

$$\dot{x}(t) = g(t, x_t) \tag{1.1}$$

where g is a suitable function defined on a domain contained in $\mathbb{R} \times C$ to $\mathbb{R}^n$, where C denotes the vector space of continuous functions from the closed interval $[-1, 0]$ to $\mathbb{R}^n$ which, endowed with the sup norm, is a Banach space. As usual, the function $x_t \in C$ is defined as

$$x_t(s) = x(t+s), \qquad -1 \le s \le 0.$$

For instance, if $g(t, \varphi) = f(t, \varphi(0), \varphi(-1))$, one obtains a DDE with constant delay 1, that is,

$$\dot{x}(t) = f(t, x(t), x(t-1)). \tag{1.2}$$

It is well known ([Hal77]) that, if g is Lipschitz, then for any initial condition $x_{t_0} \equiv u \in C$ there exists $t_f > t_0$ and a unique solution of (1.1), x^u, defined on $[t_0 - 1, t_f)$ verifying this initial condition.

It is also well known the lack of smoothness of the solution, no matter how smooth the initial data is. For instance, (1.2) can have discontinuities in the derivatives at $t_0 + k$ being k a positive integer. However, it becomes smoother in such points when k becomes larger. More precisely, if f is of class C^p and x^u is of class C^q $(q < p)$ at $t_0 + k$, it will be of class C^{q+1} at $t_0 + k + 1$. Finally, notice that if x^u is periodic, then such lack of smoothness will not appear and the periodic orbit is as smooth as f.

2 Numerical methods

There are several well-known methods for the numerical integration of DDEs (see, for instance, [BZ13]), and some are available as public domain codes. In this section we explain how to modify an existing numerical integrator (explicit or implicit) of DDEs such that it also propagates efficiently the derivatives of the flow. As an example, we have applied it to two algorithms.

The first numerical integrator has been developed by ourselves, as a modification of the standard Runge-Kutta-Fehlberg (7)8, to deal with DDEs with a single constant delay that it can be assumed to be 1. We have used Barycentric Rational Interpolation (based on the use of Chebyshev points) for the interpolation of the solution on the interval of delay. This interpolation scheme provides convergence on the whole delay interval ([BBN99]) which makes it very suitable in this situation. See Appendix A.2 for more details.

The second numerical integrator is the well known RETARD code explained in [HNW00] and provided in E. Hairer's website. It is based on a Runge-Kutta pair of orders 4 and 5 with step size control and dense output, in such a way that, after every successful integration step the computed solution is stored in a convenient way such that it is available in the next steps. Let us remark that this code can handle DDEs with several delays.

We have modified both codes to use Automatic Differentiation to propagate the derivatives of the flow, as discussed in the next section.

2.1 Automatic differentiation

Automatic Differentiation (AD) is a computational tool to obtain derivatives of the output of an algorithm w.r.t. initial data and/or parameters [GC91, Gri00, Nau12]. In this paper we are only interested in the computation of first order derivatives so we focus on this case (an extended discussion of the applications of AD with higher derivatives can be found in [JZ05, GJJC$^+$20]).

To summarize AD technique for first derivatives, assume that we have a computer program that, given a data $x \in \mathbb{R}^n$, produces a result $y \in \mathbb{R}^m$,

$$y = \Psi(x).$$

Here, Ψ denotes the implemented algorithm. Of course, we are assuming that Ψ is of class C^1. Then, given a set of data $x^0 = (x_1^0, \dots, x_n^0)$ we replace x^0 by $x^0 + s = (x_1^0 + s_1, \dots, x_n^0 + s_n)$, where s is a vector of symbols. The coefficient of the symbol s_j is the partial derivative of the actual value w.r.t x_j^0 so it is 1 at the beginning. Then, we modify the arithmetic operations in Ψ such that every operation is also performed on the first derivatives. The result of the evaluation of Ψ on x^0 is

$$x^0 \mapsto \Psi(x^0) + D\Psi(x^0)s,$$

so we have also produced the Jacobian of the function. This technique is easily implemented in C++ since this language allows for the replacement of the basic arithmetic types and operations by user defined ones. For this reason we have translated the original FORTRAN version of RETARD into C++ in order to work with our AD library.

As it has been mentioned in the Introduction, we plan to use a GMRES iterative solver. A characteristic of this method is that it requires from the user a function that, given a vector v, returns the vector Av (here A is the matrix of the linear system to be solved). In our case, this implies that we will need to compute the directional derivative of the flow (in the direction v). To do this, we will use a single symbol, say s_1, and replace the initial condition x^0 by $x^0 + s_1 v$. Using the same notation as before, and assuming that Ψ denotes the flow at a time, say, t_f, we have

$$x^0 + s_1 v \mapsto \Psi(x^0) + u s_1,$$

where u is the directional derivative of the flow in the direction v (in other words, $u = D\Psi(x^0)v$). As it has been mentioned before, we could also produce the complete Jacobian by using as many symbols as the dimension of x^0.

2.2 Poincaré sections

Given a dynamical system, a Poincaré section is defined as a codimension 1 smooth manifold transversal to the flow. In many cases, Poincaré sections are not global, in the sense that not all the trajectories of the system must cross the section. They are a standard tool to study the dynamics in regions of the space where the orbits have some kind of recurrence. In our case (the flow of a DDE), the Poincaré section is a functional space of “initial conditions” for the DDE.

In this paper we use Poincaré sections to decrease the complexity of the computations: instead of applying the Newton method to a discretization of the full periodic orbit, we apply it to the piece of trajectory contained in the section. This reduces significantly the amount of computations, specially when the period is large. To simplify the presentation, let us discuss the two kind of sections we have used.

2.2.1 Spatial section

We will focus first on the most common kind of section: the one defined by a codimension one hyperplane. They are enough for most situations and, at the same time, very easy to work with. In this paper, a C^p-section is a hyperplane

$$\mathcal{S} = \{x \in C^p([-1,0],\mathbb{R}^n) \text{ such that } \sigma(x) = 0\},$$

where $\sigma\colon C^p([-1,0],\mathbb{R}^n) \to \mathbb{R}$ is a continuous affine mapping, say $\sigma(x) = l(x) + \alpha$ such that l is a bounded linear operator and $\alpha \in \mathbb{R}$. The degree of smoothness p depends on the example at hand. As we are looking for periodic orbits, we can use the same smoothness as the DDE. Moreover, we need the flow of the DDE to be transversal to the section, at least locally. A usual transversality condition is $l(\dot{x}) > 0$. For a more detailed discussion on Poincaré sections for DDEs see, for instance, [SZ17]. To compute the

Poincaré map P, we start from some initial data in the section, we propagate the corresponding orbit till it crosses the section in the right direction and we compute the intersection point with the section. Under generic conditions, it is possible to see that the map P is continuous and compact in C^p ([SZ17]).

As it is usual in numerical methods, the elements of the Poincaré section $\mathcal{S}$ have to be discretized. Here we have used Barycentric Rational Interpolation, based on Chebyshev points. Its main advantage is that it converges on the whole interval of interpolation. See Appendix A.1 for more details. So, if these functions are discretised as a table of values $(t_i, u_i) \in [-1, 0] \times \mathbb{R}^n$, $i = 0, \dots, N$, the section is given by an expression $\sigma(u_0, \dots, u_N) = 0$, where

$$\sigma(u_0, \dots, u_N) = a + \sum_{i=0}^{N} \langle a_i, u_i \rangle \,,$$

being a and a_i fixed vectors in $\mathbb{R}^n$ defining the desired section. As usual, the integration starts from an initial condition in the section and, during the integration, the condition $\sigma(u_0, \dots, u_N) = 0$ is checked. When σ changes sign in the right direction, the equation $\sigma(u_0(t), \dots, u_N(t)) = 0$ is solved (by means of a Newton or a secant method) to find the return time to the section, $t^\star$, and the image of the initial condition by the Poincaré map.

We note that the return time $t^\star$ is not a multiple of the delay. This means that we have to interpolate the trajectory on the time interval $[t^\star - 1, t^\star]$ which contains a point on which some derivatives of the orbit are not smooth (see Section 1.1). This implies that the interpolating function does not need to be as accurate as we would expect. However, note that our goal is to compute periodic orbits, which are known to be as smooth as the DDE. In all examples considered, the DDE is C^∞ (or even analytic) so, although we may have some relevant interpolation error at the beginning of the iterations to compute the periodic orbits, this error will disappear when the iterations converge to the fixed point corresponding to a periodic orbit.

Let us comment on the computation of the differential of the Poincaré map. As it has been mentioned before, we plan to use GMRES to solve the linear systems that appear at each step of a suitable Newton method, so let us discuss first the computation of a directional derivative. In Section 2.1 we have discussed the propagation along the flow of a directional derivative by means of AD. Once the integration is finished because is has arrived to the Poincaré section at a time $t = t^\star$ (see above), the directional derivative is not contained in the tangent space of the section (in this case, as the section is an hyperplane, the tangent space is the section itself). As it is usual in this situation, the directional derivative has to be projected onto the section following the direction of the vector field of the orbit at $t = t^\star$

(the intersection point with the section). If the complete Jacobian is needed, it can be computed in the same way, applying the projection to each column.

2.2.2 Temporal section

This is the standard section used for a system that depends periodically on time, with a known period ρ. Then, generally speaking, periodic orbits have a period t^* which is a multiple of ρ. Hence, it is natural to use the Poincaré section given by the time ($t = 0 \mod \rho$) and the Poincaré map is the time-ρ flow. The differential of this Poincaré map is computed by Automatic Differentiation, as explained in Section 2.1. The computations are done in the same way as in the previous section, but note that here they are simpler: there is no need to solve an equation to arrive to the section, and the directional derivative of the flow is already the directional derivative of the map (without the need of any projection).

3 Periodic orbits

We assume that we have a DDE with a periodic orbit, with a period larger than the delay. We also assume that we have a suitable Poincaré section in the sense that the periodic orbit is transversal to it. We look for the periodic orbit as a fixed point of this Poincaré section. This means that we have to solve a nonlinear equation of the form $G(u) = \Phi(u) - u = 0$, where Φ denotes the Poincaré map. This equation is solved by Newton iteration.

3.1 The Newton method

There are two relevant ingredients in a Newton's method. The first one is to choose a suitable initial condition and the second one is to provide the differential of the function that one wants to obtain the zero. The former is usually obtained using a priory information on the model. The computation of the differential strongly depends on each situation. Clearly, the differential computation has been reduced to the computation of $D_u\Phi$. As in the ODE approach, this computation could be done by using variational equations. Here we have chosen to use Automatic Differentiation.

As we have mentioned before, the initial condition u of a DDE is discretised as a table of values, let us say (s_i, u_i), $u_i = u(s_i)$, for $i = 0, \dots, N$. After some integration steps, we obtain a new point, namely $v = x^u_{t^*+s^*}$. This numerical integration can be seen as a sequence of elementary operations that, given an initial table of values (s_i, u_i), produces another table of

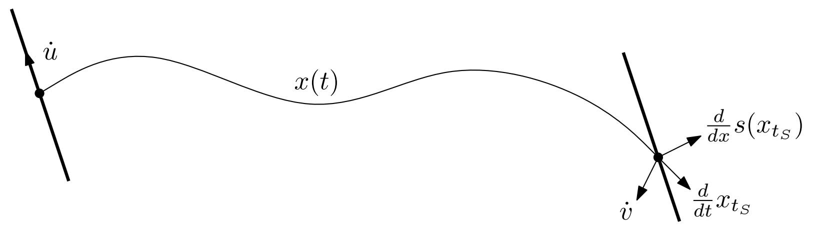

Figure 3.1. Space Poincaré section $\sigma(x)$ starting with initial symbols $\boldsymbol{u}'$. After t_S units of time the solution is crossing the section again with directional derivatives $\boldsymbol{v}'$ but it may not be in the section, so a projection to it can be done with the tangent vector $\frac{d}{dt}x_{t_S}$ and the normal to the section $\frac{d}{dx}s(x_{t_S})$.

values (s_i, v_i). Then Automatic Differentiation (AD) can be used to compute the derivatives of the result w.r.t. the initial data, by simply overloading the arithmetic used during the numerical integration. To produce the Jacobian matrix, we add a different symbol to each component of the vectors u_i, and we propagate it (to first order) along all the operations of the numerical integration. With this, we obtain the final results v_i plus linear polynomials in the initial symbols. The coefficients of these polynomials give the Jacobian of the map.

Note that we can use the same ideas to compute a directional derivative: given the initial direction, we add to the initial data a single symbol times the direction. Then, we propagate this symbol to obtain the final result plus the symbol times a vector. This vector is the directional derivative. Note that the computation of a directional derivative requires much less operations than to compute a full Jacobian.

3.1.1 Using GMRES

In some cases, the dimension of this Jacobian matrix can be large (several state variables with several delays, for instance). An option to deal with very large linear systems is to use an iterative solver like GMRES [Saa03]. These solvers do not require to obtain the full matrix of the system, they only require to compute the product of this matrix by a vector. Note that this product is the directional derivative of the Poincaré map w.r.t. this vector. Hence, instead of computing the matrix, we perform a numerical integration to compute the required directional derivatives. If the spectrum of the matrix is clustered, methods such as GMRES have a fast convergence [Bai00, DTM12]. We note that the differential of the flow of the DDEs considered here is given by a compact operator, which guarantees that the spectrum of this differential is clustered.

3.2 Stability

Once a delay periodic orbit, with initial condition u^* and period t^*, has been found by the single-shooting method explained before, its stability is determined by the spectrum of the linearised monodromy operator. According to [Hal77], if the DDE is smooth, autonomous and $t^* \geq 1$, then this linearised operator is a compact operator whose spectrum has zero as its only cluster point.

Since only a few eigenvalues are important, the computation of the respective eigenvalues and eigenfunctions should be done for some of the available methods for solving large-scale eigenvalue problems in an iterative way, in a similar way as the GMRES method. In this direction, the ARnoldi PACKage, also called ARPACK [LSY98], is capable of solving large-scale Hermitian, non-Hermitian, standard or generalised eigenvalue problems. It is designed to compute a few, say k, eigenvalues with user-specified features using $M \cdot O(k) + O(k^2)$ storage and no auxiliary extra storage required. This software is quite standard in numerical computation and it is also well-known that it is based upon an algorithmic variant of the Arnoldi process called the Implicitly Restarted Arnoldi Method (IRAM) [Sor97, LSY98].

3.3 Computer implementation

As we have already explained, Automatic Differentiation requires the use of a new arithmetic. Numerically, it can be expressed as a new data type, that is, we consider a pair (x, x') instead of just the value x where here x' denotes the derivative that is being propagated. Hence in our approach if the initial condition u of a DDE is discretised as a table of values, let us say $\boldsymbol{u} = (u_0, \ldots, u_N)$ with s_i the abscissae and $u_i = u(s_i)$ for $i = 0, \ldots, N$. To each u_i we associate a collection of symbols u'. Thus, the initial data is $(\boldsymbol{u}, \boldsymbol{u}')$ with $\boldsymbol{u}' = (u'_0, \ldots, u'_N)$. After some integration steps, we obtain a new point, namely $v = x^u_{t^*+s^*}$ which is discretised in $\boldsymbol{v}$ at the same abscissae and the propagated values $\boldsymbol{v}'$ w.r.t. the initial data $(\boldsymbol{u}, \boldsymbol{u}')$. This numerical integration can be seen as a sequence of elementary operations that, given an initial condition $(\boldsymbol{u}, \boldsymbol{u}')$, produces another value $(\boldsymbol{v}, \boldsymbol{v}')$. The size of our execution is larger because of this data type. Indeed, u_i is an n-dimensional vector and u'_i has either dimension n in the directional case or $n(N+1)$ in the Jacobian case. It is clear that this technique does not depend on the integrator of the DDE as long as it accepts the new data type and its arithmetic. Notice that, in fact, we are computing the derivative of the algorithm that computes (at the same the execution) the Poincaré map and one must be aware that if a spatial section is used, $\boldsymbol{v}'$ must be projected to the section.

4 Numerical examples

As first test, we use the model studied in [LELR97]. The equation is defined by

$$\dot{x}(t) = g(x(t-1), \alpha) = -\alpha x(t-1)\frac{1 + x(t-1)^2}{1 + x(t-1)^4}. \tag{4.1}$$

Clearly, $x \equiv 0$ is an equilibrium solution. We are going to illustrate the computation of two different cases. The first one consists in a temporal section and the second one in a spatial section.

4.1 A first example with temporal Poincaré section

Let us consider a periodic perturbation of period 2π, such as,

$$\dot{x}(t) = g(x(t-\tau), \alpha) + \varepsilon(\sin(t) + \cos(t)), \qquad \tau = 1. \tag{4.2}$$

The Implicit Function Theorem in Banach spaces tells us that there is a periodic orbit of period 2π close to $\varepsilon = 0$ and $x \equiv 0$. Thus, a temporal section of time 2π leads to a delayed stroboscopic mapping. The solution is continued using a pseudo-arc-parameter method with respect to the three parameters in (4.2), the parameter α and ε and the delay τ. In this last case, the (4.2) is modified by a change of t, that is,

$$\tau\dot{x}(t) = g(x(t-1), \alpha) + \varepsilon(\sin(\tau t) + \cos(\tau t)). \tag{4.3}$$

Because of the change of time, the temporal section is also suitably modified with the value of $2\pi/\tau$.

The initial values of these parameters has been $\alpha = 1.5$, $\varepsilon = 10^{-4}$ and $\tau = 1$ with a discretisation size $N = 32$. Figure 4.1 show the results, the stability and the different bifurcations. Notice that in the continuation of ε a symmetry with respect to the y-axis holds.

4.2 An example with a spatial Poincaré section

Studying the stability of the equilibrium point of the system (4.1) with respect to the parameter, one shows a family of periodic solutions bifurcates at $\alpha = \frac{\pi}{2}$ because of a Hopf bifurcation. These periodic orbits have an unknown period and a spatial section will allow us to compute one of them. If the

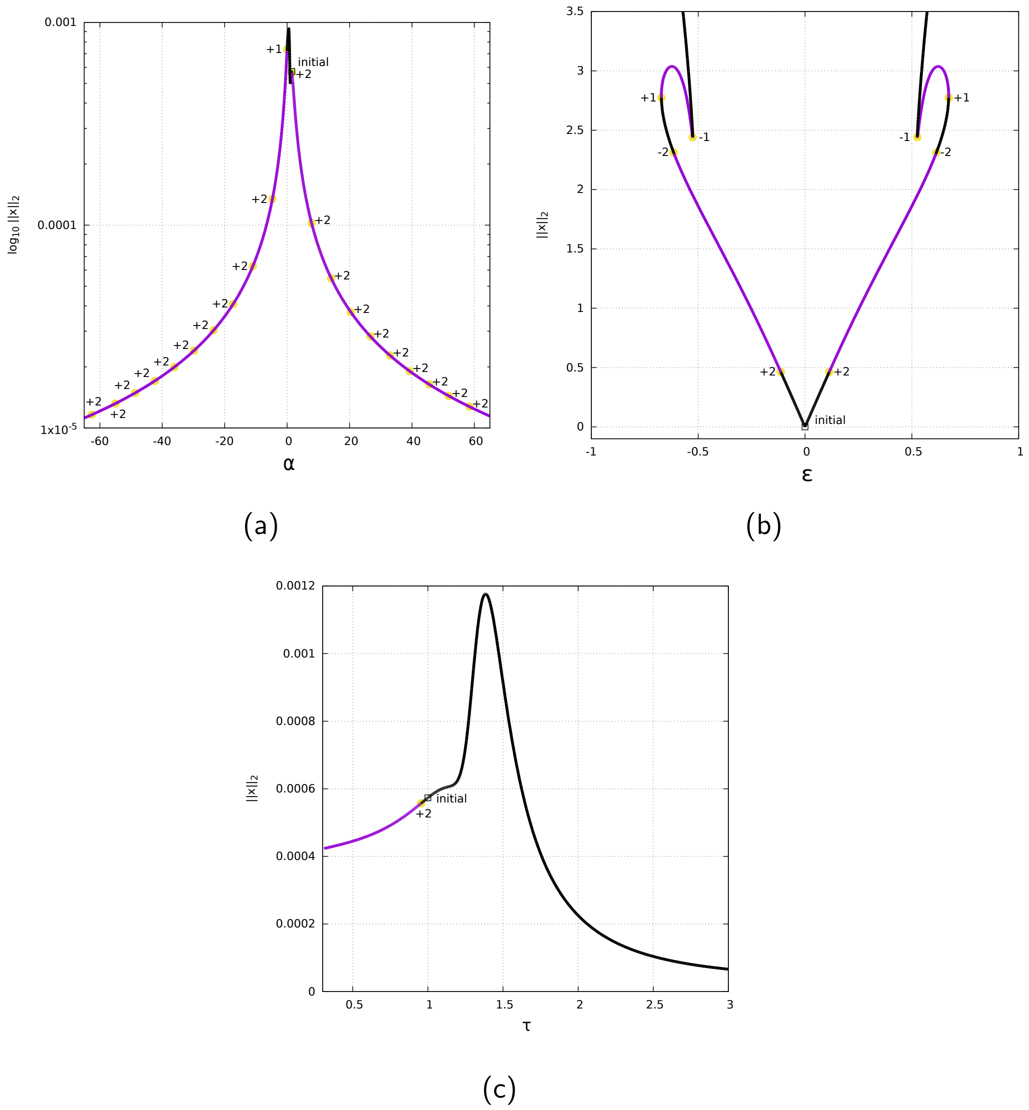

Figure 4.1. Continuation of periodic orbits of (4.3) with respect to parameters; (a) has $\varepsilon = 10^{-4}$ and $\tau = 1$, (b) has $\alpha = 1.5$ and $\tau = 1$, and (c) has $\alpha = 1.5$ and $\varepsilon = 10^{-4}$. Black colour means stable.

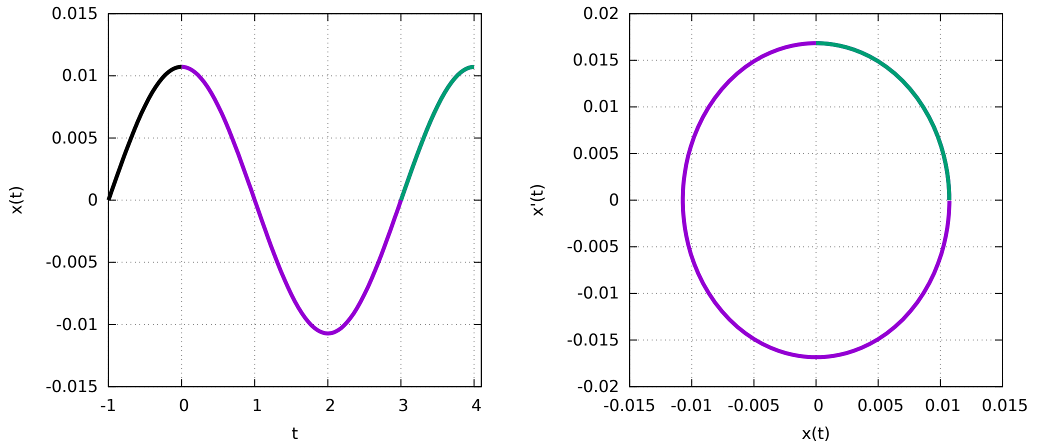

Figure 4.2. Periodic orbit of the equation (4.1). In the left hand side, the orbit is displayed with the initial condition in $-1 \leq t \leq 0$ and final lag-segment once the second has been crossed two times. The phase space of the periodic orbit is shown in the right hand side.

initial condition is discretised by a table of values (s_i, u_i), $i = 0, \ldots, N$. The section is given by an affine hyperplane, let us say,

$$\sigma(x_0, \ldots, x_N) = a + \sum_{i=0}^{N} a_i x_i.$$

The values a and a_i must be fixed at each step of the continuation in such a way that the section must be transverse to the flow. We have selected the easiest one $\sigma(x_0, \ldots, x_N) = x_0$ and α close to the bifurcation point, e.g. $\alpha = 1.57$. Figure 4.2 shows the periodic orbit around the solution $x \equiv 0$ and $\alpha = 1.57$. Such a computation has been done with a Newton tolerance of 10^{-12}, a Section tolerance 10^{-14} and an integrator tolerance of 10^{-15}. Once an initial periodic orbit has been computed for $\alpha = 1.57$ a pseudo-arc-length method ([Sim90]) can be done to perform a continuation with respect to the parameter α, see Figure 4.3. Now the propagated derivatives must be modified in such a way that they are also in the spatial section. But the row corresponding to the new equation must not. Additionally, one can also show how the periodic orbits evolve with respect to the parameter, Figure 4.4. Notice that each of those periodic orbits have period 4 in the main branch.

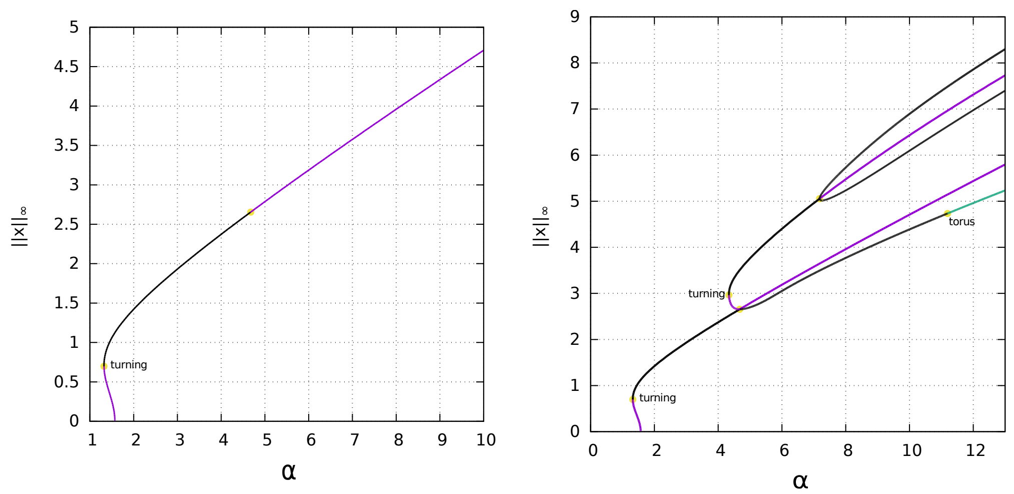

Figure 4.3. Continuation of periodic orbits with respect to α starting at the periodic orbit computed for $\alpha = 1.57$, $\varepsilon = 10^{-4}$ and $\tau = 1$ of Equation (4.1). The y-axis represents the ∞-norm of the initial condition of each of the periodic orbits. Black colour means stable.

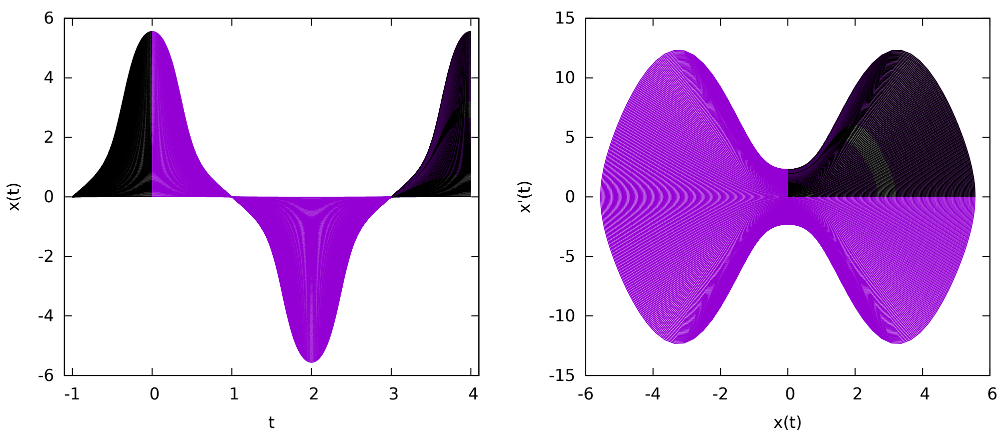

Figure 4.4. Same continuation as in Figure 4.3, left. Left plot: positions vs. time. Right plot: derivatives vs. positions.

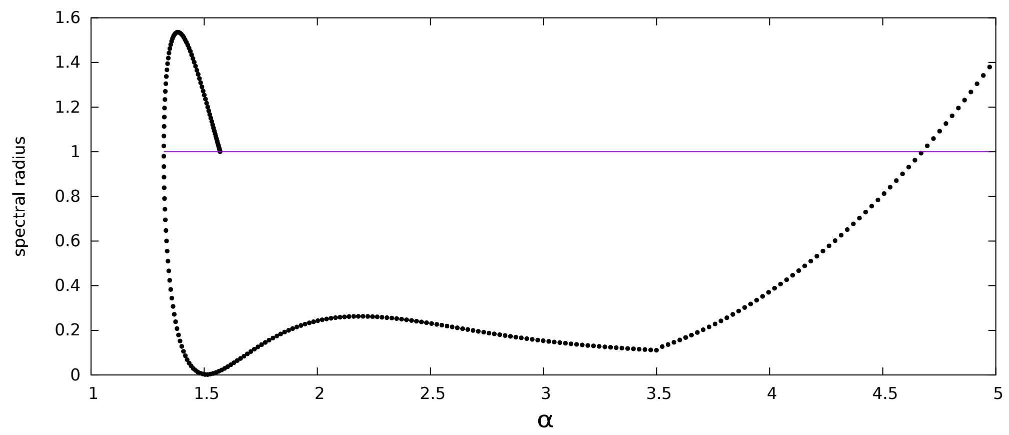

Figure 4.5. Same continuation as in Figure 4.3, left, now showing the evolution of the spectral radius. We have added a horizontal straight line at 1 to visualise the changes of stability.

4.2.1 Computing the eigenvalues

Once a delay periodic orbit has been found, the eigenpairs of the delay discrete Poincaré map may show information of its stability. Indeed, Figure 4.5 displays the modulus of the largest eigenvalue over the continuation of the periodic orbit through the parameter α in (4.1). Notice that the continuation done in the Figure 4.3 reaches values of α higher than 5, but for $\alpha > 5$ the spectral radius remains bigger than 1. The "corner" close to $\alpha = 3.5$ in Figure 4.5 arises because the second eigenvalue with largest modulus becomes the first one.

As a final comment, the CPU time so as to compute Figures from 4.2 to 4.5 has been around 11 minutes counting the I/O functions (i.e. prints, read/write by stream, ...).

4.3 An example with several delays

The next example illustrates that our method is independent of the model whenever one has a suitable integrator. In particular, it can be applied for several delays, such as, the equation

$$\dot{x}(t) = -(\lambda_1 x(t-\tau_1) + \lambda_2 x(t-\tau_2) + \lambda_3 x(t-\tau_3))(1 + x(t)). \tag{4.4}$$

It has been studied extensively [Nus78, KL12] and existence of periodic orbits has been proved for some values of the parameters. We take concrete values: $\lambda_1 = \lambda_2 = 2.5$, $\lambda_3 = 0.25$, $\tau_1 = 1.65$, $\tau_2 = 0.35$ and $\tau_3 = 1$ such that the existence of a periodic orbit is guaranteed. The initial condition u at time

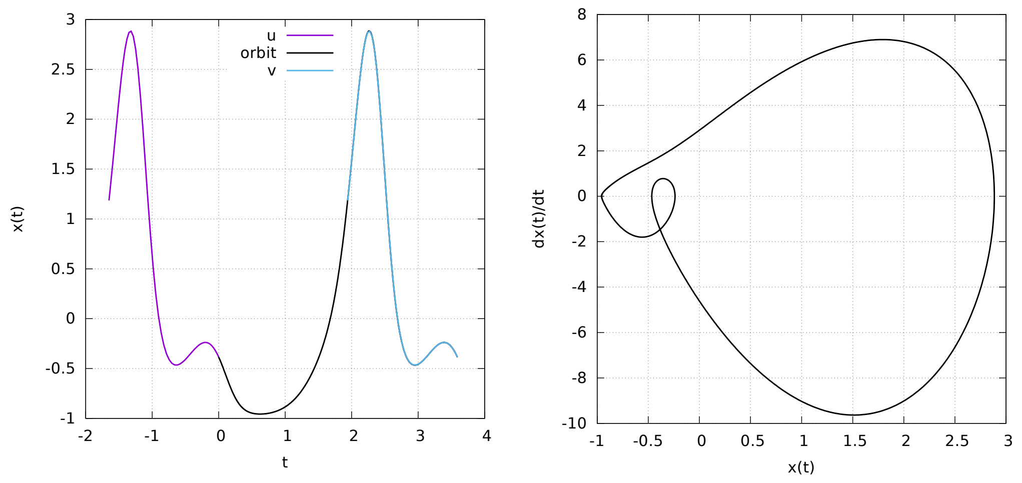

Figure 4.6. Periodic orbit of (4.4) with parameters $\lambda_1 = \lambda_2 = 2.5$, $\lambda_3 = 0.25$, $\tau_1 = 1.65$, $\tau_2 = 0.35$ and $\tau_3 = 1$. The final period is almost 3.5894 units of time.

$t_0 = 0$ is the integration of the orbit through $\cos(t)$ at $t_0 = 0$ until a final time of 4. This initial condition u is discretised with $N = 64$, i.e. 65 points, and (s_i, u_i) with s_i Chebyshev abscissae of second type and $u_i = u(s_i)$. We have applied the spatial section $x_N = u_N$ to detect a periodic orbit after 4 crosses using the Hairer's integrator [HNW00] with a tolerance 10^{-14}. In this case, we have executed the two approaches of AD, that is, a full Jacobian implementation, that uses an LU solver, and a Jacobian-free implementation that uses a GMRES solver. The maximum dimension of the Krylov subspace has been 5 for each of the six different linear systems to be solved during the Newton's method with a tolerance of 10^{-12}. The use of the iterative solver allows us to consider $10^{-3}\|\boldsymbol{b}\|_2$ as stopping criterion of the GMRES of the linear system $A\boldsymbol{x} = \boldsymbol{b}$. The CPU time to compute the periodic orbit is 4 seconds in the case of the full Jacobian while the Jacobian-free only needs 0.73 seconds. The Figure 4.6 shows the orbit which is stable after computing the three first eigenvalues.

5 Conclusions and future work

This paper illustrates, in essence, that the same techniques applied in ordinary differential equations to compute a periodic orbit and to perform a continuation with respect to parameters can be done in a delay differential equation context. Although these techniques are almost the same, the main contribution is how the differential required by the Newton's method is computed. This is done by changing (enlarging) the standard double precision type to a new data type that also contains derivatives (which are also double precision types). The computer arithmetic is then enlarged to operate on this new data type so that it propagates not only values but also derivatives along the algorithms (this is done quite easily in languages like C++). The advantage of this point of view is that these techniques can be applied very easily to any numerical integrator.

We note that, to compute a periodic orbit, we only need to discretise a segment of the orbit (of the size of the delay), which implies that the number of unknowns of the linear system is smaller than the number of unknowns obtained when discretising the full orbit. Moreover, when the problem becomes more complex and the Newton method leads to high dimensional systems, we have shown that iterative methods such as GMRES work very effectively (see Section 4.3). Finally, we have seen that the stability of the orbits can also be studied efficiently by means of iterative methods such as those based on Arnoldi's schemes.

To illustrate these methods, we have considered several DDEs models with families of periodic orbits and Hopf bifurcations. We have continued these families and computed their stability. In the near future we expect to extend the methods being used for quasi-periodic solutions in ODEs to DDEs by means of similar techniques.

A Another delay time integrator

Let us assume a Delay Differential Equation (DDE) with constant delay. It can be seen as a chain of Ordinary Differential Equations (ODEs). Therefore a reasonable first approach is to consider a standard integrator and whenever an unknown value was required, it would be interpolated. However, such an interpolation should be as far accurate as the integrator is. We propose a Runge-Kutta-Fehlberg as ODE integrator and barycentric rational interpolation.

A.1 Barycentric Rational Approximation

Let $\varphi\colon [a,b] \to \mathbb{C}$ be a function, ϕ be the affinity defined by

$$\begin{aligned}\phi\colon [-1,1] &\longrightarrow [a,b] \\ x &\longmapsto \frac{a-b}{2}x + \frac{a+b}{2} \\ \frac{2y-a-b}{a-b} &\longleftarrow y.\end{aligned} \tag{A.1}$$

and $x_0,\dots,x_N$ be distinct points in $[-1,1]$. The Barycentric Rational Interpolation with weights $w_0,\dots,w_N$ approximates φ via the formula

$$R[\varphi;x_k](y) = \frac{\displaystyle\sum_{k=0}^{N} \frac{w_k}{x-x_k}\varphi_k}{\displaystyle\sum_{k=0}^{N} \frac{w_k}{x-x_k}}, \qquad \phi(x) = y \tag{A.2}$$

with $\varphi_k = (\varphi\circ\phi)(x_k)$. Clearly, the computational and space complexities of (A.2) are linear on N.

If x_k are the Chebyshev points of first type defined by

$$\mathtt{I}_k = \mathtt{I}_k^N := \cos\left(\frac{\pi(k+\frac{1}{2})}{N}\right), \qquad 0 \le k < N$$

N.B.: $\mathtt{I}_0 > \cdots > \mathtt{I}_{N-1}$ and $\mathtt{I}_k = -\mathtt{I}_{N-1-k}$ for any $0 \le k < \lceil \frac{N}{2} \rceil$.

and N is even, (A.2) becomes

$$R[\varphi;\mathtt{I}](y) = \frac{\displaystyle\sum_{0\le k<\frac{N}{2}} (-1)^k \frac{\mathtt{I}_{N/2-1-k}}{x-\mathtt{I}_k}\varphi_k + \sum_{\frac{N}{2}\le k<N} (-1)^k \frac{\mathtt{I}_{k-N/2}}{x+\mathtt{I}_{N-1-k}}\varphi_k}{\displaystyle\sum_{0\le k<\frac{N}{2}} (-1)^k \frac{\mathtt{I}_{N/2-1-k}}{x-\mathtt{I}_k} + \sum_{\frac{N}{2}\le k<N} (-1)^k \frac{\mathtt{I}_{k-N/2}}{x+\mathtt{I}_{N-1-k}}}.$$

Therefore it is enough to store the first $\frac{N}{2}$ Chebyshev points and the N function values.

On the other hand, if x_k are the Chebyshev points of second type defined by

$$\mathtt{II}_k = \mathtt{II}_k^N := \cos\left(\frac{\pi k}{N}\right), \qquad 0 \le k \le N,$$

N.B.: $\mathtt{II}_0 > \cdots > \mathtt{II}_N$ and $\mathtt{II}_k = -\mathtt{II}_{N-k}$ for any $0 \le k < \lceil \frac{N}{2} \rceil$.

(A.2) becomes

$$R[\varphi;\mathtt{II}](y) = \frac{\frac{\frac{1}{2}\varphi_0}{x-\mathtt{II}_0} + \sum\limits_{1\le k\le \frac{N}{2}} \frac{(-1)^k\varphi_k}{x-\mathtt{II}_k} + \sum\limits_{\frac{N}{2}<k<N} \frac{(-1)^k\varphi_k}{x+\mathtt{II}_{N-k}} + \frac{(-1)^N\frac{1}{2}\varphi_N}{x+\mathtt{II}_0}}{\frac{\frac{1}{2}}{x-\mathtt{II}_0} + \sum\limits_{1\le k\le \frac{N}{2}} \frac{(-1)^k}{x-\mathtt{II}_k} + \sum\limits_{\frac{N}{2}<k<N} \frac{(-1)^k}{x+\mathtt{II}_{N-k}} + \frac{(-1)^N\frac{1}{2}}{x+\mathtt{II}_0}}.$$

So it is enough to store the first $\frac{N}{2}+1$ Chebyshev points and the $N+1$ function values. In both cases, $R[\varphi;\mathtt{I}]$ and $R[\varphi;\mathtt{II}]$ does not have any pole in $[-1,1]$. Furthermore, under analytic hypothesis one proves that the error with respect to the exact value is $O(\rho^{-N})$ for some $\rho>0$, [BBN99]. The error behaviour of each of them are really similar although $R[\varphi;\mathtt{I}]$ has an error more similar to that computed by the classical Chebyshev approximation. Hence, one may think that it does not matter whether I or II are used but each of them have their advantages. For instance,

- I does not contain the values -1 and 1.
- II contains the values -1 and 1.
- $\mathtt{II}_{ak}^{aN} = \mathtt{II}_k^N$ for any $0\le k\le N$ and positive integer a.
- If N is even, then $\mathtt{II}_{N/2}=0$.

Notice also that if φ gives values on an n-dimensional space, one can apply the same scheme in each of the n coordinates so the complexity will be $\Theta(nN)$.

A.2 Delay-Runge-Kutta-Fehlberg

The scheme stores the values at the Chebyshev points in the lag interval $I_k = [t_0+k, t_0+(k+1)]$, it computes and stores the values at the Chebyshev points of the next lag interval $I_{k+1}=[t_0+(k+1), t_0+(k+2)]$ and the evaluation of the DDE is done by the Chebyshev approximation or the Barycentric Rational Approximation on I_k. Let us assume that the integration step is performed by a Runge-Kutta-Fehlberg method, namely `rkf`. The Algorithm 1 sketches the method. Note that the main idea is that the step h, which is modified during the integration, must be bounded with a maximum step whose value is just the difference between Chebyshev points, the last point of the interval or the final time.

Input: t_0, u initial condition, final time t_f and initial step h.
Output: The solution value at the final time.

```
1:  $a \leftarrow t_0 - \tau \quad b \leftarrow t_0 \quad \alpha \leftarrow \frac{1}{2}(a-b) \quad \beta \leftarrow \frac{1}{2}(a+b)$
2:  for $k = 0$ to $N-1$ do
3:      $p_k \leftarrow u(\alpha \mathtt{I}_k + \beta)$
4:  $h_{\min} \leftarrow 10^{-9}$
5:  $t \leftarrow t_0$
6:  while $t < t_f$ do
7:      $b \leftarrow b + \tau$
8:      $\beta \leftarrow \beta + \tau$
9:      for $k = 0$ to $N-1$ do
10:         $\gamma \leftarrow \min\{\alpha \mathtt{I}_k + \beta, t_f\}$
11:         $h_{\max} \leftarrow \gamma - t$
12:         $h \leftarrow \frac{h}{|h|} \min\{|h|, h_{\max}\}$
13:         $\mathtt{rkf}(\gamma, t, h, x, p, \tau)$
14:         if $t = t_f$ then
15:             return
16:         $c_k \leftarrow x$
17:     $\gamma \leftarrow \min\{b, t_f\}$
18:     $h_{\max} \leftarrow \gamma - t$
19:     $h \leftarrow \frac{h}{|h|} \min\{|h|, h_{\max}\}$
20:     $\mathtt{rkf}(\gamma, t, h, x, p, \tau)$
21:     $p \leftrightarrow c$
```

Algorithm 1. Delay-Runge-Kutta-Fehlberg integrator scheme.

References

[Bai00] Z.-Z. Bai. Sharp error bounds of some Krylov subspace methods for non-Hermitian linear systems. *Appl. Math. Comput.*, 109(2-3):273–285, 2000.

[BBN99] R. Baltensperger, J.-P. Berrut, and B. Noël. Exponential convergence of a linear rational interpolant between transformed Chebyshev points. *Math. Comp.*, 68(227):1109–1120, 1999.

[BZ13] A. Bellen and M. Zennaro. *Numerical methods for delay differential equations*. Numerical Mathematics and Scientific Computation. Oxford University Press, Oxford, 2013. First paperback reprint of the 2003 original [MR1997488].

[DTM12] J. Duintjer Tebbens and G. Meurant. Any Ritz value behavior is possible for Arnoldi and for GMRES. *SIAM J. Matrix Anal. Appl.*, 33(3):958–978, 2012.

[GC91] A. Griewank and G.F. Corliss, editors. *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*. SIAM, Philadelphia, Penn., 1991.

[GJJC$^+$20] J. Gimeno, À. Jorba, M. Jorba-Cuscó, N. Miguel, and M. Zou. Numerical integration of high order variational equations of ODEs. Preprint, 2020.

[Gri00] A. Griewank. *Evaluating Derivatives*. SIAM, Philadelphia, Penn., 2000.

[Hal77] J. Hale. *Theory of functional differential equations*. Springer-Verlag, New York-Heidelberg, second edition, 1977. Applied Mathematical Sciences, Vol. 3.

[HMN91] Y. Hino, S. Murakami, and T. Naito. *Functional-differential equations with infinite delay*, volume 1473 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1991.

[HNW00] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving ordinary differential equations I. Nonstiff problems*, volume 8 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second revised edition, 2000.

[JZ05] À. Jorba and M. Zou. A software package for the numerical integration of ODEs by means of high-order Taylor methods. *Exp. Math.*, 14(1):99–117, 2005.

[KL12] G. Kiss and J.-P. Lessard. Computational fixed-point theory for differential delay equations with multiple time lags. *J. Differential Equations*, 252(4):3093–3115, 2012.

[LELR97] T. Luzyanina, K. Engelborghs, K. Lust, and D. Roose. Computation, continuation and bifurcation analysis of periodic solutions of delay differential equations. *Internat. J. Bifur. Chaos Appl. Sci. Engrg.*, 7(11):2547–2560, 1997.

[LSY98] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK users' guide*, volume 6 of *Software, Environments, and Tools*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1998. Solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods.

[Nau12] U. Naumann. *The art of differentiating computer programs*, volume 24 of *Software, Environments, and Tools*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2012. An introduction to algorithmic differentiation.

[Nus78] R.D. Nussbaum. Differential-delay equations with two time lags. *Mem. Amer. Math. Soc.*, 16(205):vi+62, 1978.

[Saa03] Y. Saad. *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, second edition, 2003.

[SEL$^+$15] J. Sieber, K. Engelborghs, T. Luzyanina, G. Samaey, and D. Roose. DDE-BIFTOOL v. 3.0 Manual - Bifurcation analysis of delay differential equations. *arXiv:1406.7144*, 125:265–275, 2015.

[Sim90] C. Simó. On the analytical and numerical approximation of invariant manifolds. In D. Benest and C. Froeschlé, editors, *Modern methods in celestial mechanics*, pages 285–330. Ed. Frontières, 1990.

[Sor97] D. C. Sorensen. Implicitly restarted Arnoldi/Lanczos methods for large scale eigenvalue calculations. In *Parallel numerical algorithms (Hampton, VA, 1994)*, volume 4 of *ICASE/LaRC In-*

terdiscip. Ser. Sci. Eng., pages 119–165. Kluwer Acad. Publ., Dordrecht, 1997.

[SW06] A.L. Skubachevskii and H.-O. Walther. On the Floquet multipliers of periodic solutions to non-linear functional differential equations. *J. Dynam. Differential Equations*, 18(2):257–355, 2006.

[SZ17] R. Szczelina and P. Zgliczyński. Algorithm for rigorous integration of Delay Differential Equations and the computer-assisted proof of periodic orbits in the Mackey-Glass equation. *Found. Comput. Math.*, 18(6):1299–1332, 2017.