

**A Newton-Picard collocation method for  
periodic solutions of delay differential  
equations**

*Koen Verheyden*      *Kurt Lust*

*Report TW 357, April 2003*



**Katholieke Universiteit Leuven**  
**Department of Computer Science**

Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

# A Newton-Picard collocation method for periodic solutions of delay differential equations \*

*Koen Verheyden*      *Kurt Lust*

*Report TW 357, April 2003*

Department of Computer Science, K.U.Leuven

## Abstract

This paper presents a collocation method with an iterative linear system solver to compute periodic solutions of a system of autonomous delay differential equations (DDEs). We show that the linearized collocation system is equivalent to a discretization of the linearized periodic boundary value problem (BVP). This linear BVP is solved using the Newton-Picard single shooting method ([Int. J. Bifurcation Chaos, 7 (1997), pp. 2547–2560]). The Newton-Picard method combines a direct method in the subspace of the weakly stable and unstable modes with an iterative solver in the orthogonal complement. As a side effect, we also obtain good estimates for the dominant Floquet multipliers. We have implemented the method in the DDE-BIFTOOL environment to test our algorithm.

**Keywords** : delay differential equation, periodic solution, Newton-Picard, collocation, numerical bifurcation analysis

**AMS(MOS) Classification** : 65J15, 65P30, 65Q05

---

\* Submitted to BIT.

# 1 Introduction.

In this paper, we present a new method to compute periodic solutions of a system of autonomous *delay differential equations* (DDEs),

$$x'(t) = f(x(t), x(t - \tau_1), \dots, x(t - \tau_\kappa), \gamma), \quad (1)$$

where  $x(t) \in \mathbb{R}^n$  and  $\tau_k$  for  $k = 1, \dots, \kappa$  are constant, non-negative delays. The vector  $\gamma \in \mathbb{R}^L$  groups a number of (physical) parameters, and  $f$  is a two times continuously differentiable function from  $\mathbb{R}^n \times \mathbb{R}^{n \times \kappa} \times \mathbb{R}^L$  into  $\mathbb{R}^n$ .

We search for periodic orbits  $x^*(t)$  of (1), i.e., orbits that are non-constant (i.e., not an equilibrium point) and for which

$$x^*(t + T^*) = x^*(t), \quad \text{for all } t \in \mathbb{R}, \quad (2)$$

for some non-zero  $T^*$ . The smallest strictly positive value for  $T^*$  is called the *period*. Several techniques have been proposed to solve this BVP.

Single shooting is probably the simplest technique. It can be implemented on top of an existing time integration code. This method is used in [18], where it is combined with the Newton-Picard method. The latter method was first developed to compute periodic solutions of parabolic PDEs ([19, 15, 16]). The method is designed to be efficient when only a few Floquet multipliers are close to or outside the unit circle. It combines a direct linear solver in the generalized eigenspace of these dominant Floquet multipliers with an iterative method equivalent to time integration in the orthogonal complement. It also computes the dominant Floquet multipliers.

Collocation methods are a more robust alternative to shooting methods. The ODE bifurcation package AUTO ([5, 6]) and the package DDE-BIFTOOL for the bifurcation analysis of DDEs ([11, 13, 12]) are both based on Runge-Kutta collocation. A disadvantage of DDE-BIFTOOL's implementation is the high computational and memory cost for solving the linearized systems. Though these systems are usually sparse, it is often impossible to exploit the structure in a direct method. This is different from the ODE case. In AUTO, e.g., a special-purpose linear system solver is used to exploit the structure of the linearized collocation system. Moreover, the Floquet multipliers are easily computed from two submatrices appearing in one of the final steps of that solver. In DDE-BIFTOOL, the periodic orbit is computed first. To compute the Floquet multipliers afterwards, a slightly different collocation discretization is used, resulting in a much larger, but more structured, linear system. The monodromy matrix is captured by condensation of the Jacobian matrix. Our method will start from this collocation scheme. It is easily shown that the linearized collocation system is also the discretization of a BVP for the linearized DDE system, i.e., linearization and discretization “commute”. This linear BVP can be solved using the Newton-Picard method. In doing so, we obtain an algorithm combining the main advantages of both methods (the robustness of a collocation method and the efficiency of the single shooting Newton-Picard method) without the disadvantages.

The plan of the paper is as follows. In section 2, the periodic BVP and the monodromy operator are introduced. In section 3 we describe our Runge-Kutta collocation scheme as well as the linearized BVP. Next, we discuss the solution of this BVP with single shooting in section 4. We also show how time integration of the linearized DDE and matrix-vector products with the monodromy matrix can be computed efficiently, relying on information from the linearized collocation system. This is the key to the application of the Newton-Picard method in section 5. Compared to earlier work ([15, 16]) several extensions were made to the Newton-Picard method to increase the robustness and the efficiency. We made sure that the resulting algorithm behaves as closely as possible like the collocation scheme in DDE-BIFTOOL rather than like a single shooting method. In section 6, the

Newton-Picard collocation method is extended to continuation. In section 7 we present results for periodic solutions of two models. Finally, section 8 summarizes the main conclusions from this work.

## 2 Periodic solutions of DDEs.

Consider again the DDE system (1) and define  $\tau := \max_k \tau_k$ , the maximal delay. For  $t_o \in \mathbb{R}$ ,  $x_{t_o}$  denotes a *segment* of the function  $x(t)$ , defined by

$$x_{t_o}(\theta) = x(t_o + \theta), \quad \text{for } \theta \in [-\tau, 0].$$

This function segment  $x_{t_o}$  uniquely determines  $x(t)$  for all  $t \geq t_o$ . Hence, the state space for (1) is

$$\mathcal{C}_\tau := \mathcal{C}([-\tau, 0], \mathbb{R}^n),$$

the vector space of all continuous functions mapping the interval  $[-\tau, 0]$  into  $\mathbb{R}^n$ . This implies that the periodicity condition (2) in the DDE case is equivalent to the equality of the *initial function segment*  $x_0$  and the *final function segment*  $x_T$  of  $x(t)$ , i.e.,

$$x_T(\theta) = x_0(\theta), \quad \text{for } \theta \in [-\tau, 0], \quad (3)$$

instead of just  $x(T) = x(0)$  in the ODE case. We will now first discuss the periodic BVP that determines the periodic orbit in a unique way. Next, in section 2.2, we will discuss the stability of the orbit.

### 2.1 The boundary value problem.

Before computing a periodic orbit of (1), we introduce a scaled time variable,  $s := t/T$  and substitute  $y(s) := x(t)$ . Let  $\tau_0 := 0$  for notational convenience and denote by  $\mathcal{F}$  the operator that maps  $\mathcal{C}([-\tau/T, 1], \mathbb{R}^n) \times \mathbb{R}$  into  $\mathcal{C}([0, 1], \mathbb{R}^n)$  defined by

$$\mathcal{F}(y, T)(s) := Tf(y(s - \tau_0/T), \dots, y(s - \tau_\kappa/T)), \quad \text{for } s \in [0, 1]. \quad (4)$$

Since we are considering the computation of a single periodic orbit in most of this paper, the fixed parameter  $\gamma$  is dropped from our notations. Because we are dealing with autonomous systems, (1) together with (3) is a sufficient condition for a periodic orbit ([11]). However, one has to add another constraint to obtain isolated solutions. The resulting BVP, using the rescaled time  $s$  and with unknowns  $(y, T)$ , is

$$y'(s) = \mathcal{F}(y, T)(s), \quad \text{for } s \in [0, 1], \quad (5a)$$

$$y_1(\vartheta) - y_0(\vartheta) = 0, \quad \text{for } \vartheta \in [-\tau/T, 0], \quad (5b)$$

$$\alpha(y) = 0. \quad (5c)$$

Obviously, the state space for the rescaled DDE system (5a) is  $\mathcal{C}_{\tau/T} := \mathcal{C}([-\tau/T, 0], \mathbb{R}^n)$ . Hence, the *initial function segment*  $y_0$  and *final function segment*  $y_1$  of  $y(s)$  belong to  $\mathcal{C}_{\tau/T}$ . The *phase condition* (5c) is used to remove translational invariance. We choose the phase condition

$$\alpha(y) := \int_0^1 y^T(s) \frac{d}{ds} y^\circ(s) ds = 0, \quad (6)$$

which minimizes the phase shift of  $y(s)$  with respect to some reference solution ( $y^\circ(s) \equiv x^\circ(sT^\circ), T^\circ$ ). This condition is also used in AUTO ([7]). It is a very good condition in combination with adaptive meshes since it reduces the need for frequent remeshing.

## 2.2 The monodromy operator.

Let us define the *time integration operator*  $\mathcal{T}$  that maps  $\mathcal{C}_{\tau/T}$  into  $\mathcal{C}_{\tau/T}$  such that  $\mathcal{T}(y_s, \Delta s) = y_{s+\Delta s}$ . For a periodic solution  $(y^*, T^*)$ , where  $\mathcal{T}(y_0^*, 1) = y_1^* = y_0^*$ , the *monodromy operator*  $\mathcal{M}^*(y_0^*)$  is defined as the Fréchet derivative of the time integration operator at  $\Delta s = 1$  ([8, 20]), i.e.,

$$\mathcal{M}^*(y_0^*) := \frac{\partial \mathcal{T}(y_0^*, 1)}{\partial y_0^*}. \quad (7)$$

Hence, the function segment  $y_0^* + \mathcal{M}^*(y_0^*)\Delta y_0$  is, up to first order terms, the image of the perturbed function segment  $y_0^* + \Delta y_0$  under integration of (5a) over an interval of length  $\Delta s = 1$ . An alternative — but equivalent ([4]) — definition of the monodromy operator characterizes  $\mathcal{M}^*(y_0^*)\Delta y_0$  as the function segment in  $\mathcal{C}_{\tau/T}$  that results from integration of the *variational equations* of (5a) about the periodic solution  $(y^*, T^*)$ ,

$$\begin{aligned} (\Delta y)'(s) &= \mathcal{F}_y(y^*, T^*)\Delta y(s) \\ &= T \sum_{k=0}^{\kappa} \frac{\partial f}{\partial y^k}(y^0, \dots, y^\kappa) \Big|_{(y^*(s), \dots, y^*(s-\tau_k/T))} \Delta y(s - \tau_k/T), \end{aligned} \quad (8)$$

over  $\Delta s = 1$ , starting from the initial function segment  $\Delta y_0 \in \mathcal{C}_{\tau/T}$ . Note that the Fréchet derivative  $\mathcal{F}_y(y^*, T^*)$  maps  $\mathcal{C}([-\tau/T, 1], \mathbb{R}^n)$  into  $\mathcal{C}([0, 1], \mathbb{R}^n)$ .

The monodromy operator  $\mathcal{M}^*(y_0^*)$  and its eigenfunctions depend on the choice of the position of the function segment  $y_0^*$  on the periodic orbit  $y(s)$ , but the spectrum is independent of that choice ([14, 4]). The nonzero points in the spectrum are isolated eigenvalues with finite multiplicity, called *Floquet multipliers* and denoted by  $\mu^*$ . There is always a *trivial Floquet multiplier* equal to one, corresponding to a perturbation along the periodic orbit. A periodic solution  $(y^*, T^*)$  is (asymptotically) stable if all Floquet multipliers, except the trivial one, lie strictly within the unit circle in the complex plane.

For practical purposes, the definition of the monodromy operator is extended to the case where  $(y, T)$  is an *approximate* solution to (5). This is denoted by removing the “\*”. The action of the monodromy operator is then obtained by integration of (8) about the orbit of (5a) with initial condition  $\Delta y_0$ .

We also define the operator  $\mathcal{M}_e(y_0)$  that maps  $\mathcal{C}_{\tau/T}$  into  $\mathcal{C}([0, 1], \mathbb{R}^n)$  such that  $\mathcal{M}_e(y_0)\Delta y_0$  gives the solution of the variational equations (8) on the interval  $]0, 1]$  for a given initial function segment  $\Delta y_0$  on  $[-\tau/T, 0]$ .

## 3 Discretization and linearization.

In section 3.1 and 3.2, we describe our particular Gauss-Legendre Runge-Kutta collocation variant. Next, in section 3.3, we develop an alternative view on the resulting linearized system. This will motivate the construction of the monodromy matrix, the discretization of the monodromy operator, in section 3.4.

### 3.1 Gauss-Legendre Runge-Kutta collocation.

Let  $\Sigma^m := \{0 = s_0 < s_1 < \dots < s_m = 1\}$  be a mesh on  $[0, 1]$ . This mesh is periodically extended to the left to obtain a mesh  $\Sigma_{-\ell}^m := \{s_{-\ell}, \dots, s_m = 1\}$  on  $[s_{-\ell}, 1] \supseteq [-\tau/T, 1]$ . We determine  $\ell$  such that

$$s_{-\ell} \leq -\tau/T < s_{-\ell+1}. \quad (9)$$

Denote by  $\mathbb{P}_d^n|_D$  the set of polynomials of degree  $d$  or less that map  $D \subseteq \mathbb{R}$  into  $\mathbb{R}^n$ . We approximate a function  $y \in \mathcal{C}([s_{-\ell}, 1], \mathbb{R}^n)$  by an element  $u$  of the space of piecewise polynomials

$$\mathbb{P}_d^n(\Sigma_{-\ell}^m) := \left\{ u \in \mathcal{C}([s_{-\ell}, 1], \mathbb{R}^n) : u|_{[s_i, s_{i+1}]} \in \mathbb{P}_d^n|_{[s_i, s_{i+1}]}, i = -\ell, \dots, m-1 \right\}.$$

We will represent the elements of this space in a convenient way. Let  $\Delta s_i := s_{i+1} - s_i$  and  $s_{i+\frac{j}{d}} := s_i + \frac{j}{d}\Delta s_i$  for  $i = -\ell, \dots, m-1$  and  $j = 0, \dots, d$ . It is possible to choose scalar basis functions  $\phi_{i+\frac{j}{d}} \in \mathbb{P}_d^1(\Sigma_{-\ell}^m)$  that satisfy  $\phi_{i+\frac{j}{d}}(s_{i+\frac{\hat{j}}{d}}) = \delta_{i,\hat{i}}\delta_{j,\hat{j}}$ , for  $i, \hat{i} = -\ell, \dots, m-1$  and  $j, \hat{j} = 0, \dots, d$ , with  $\delta_{i,j}$  the *Kronecker delta*. Hence, a piecewise polynomial  $u \in \mathbb{P}_d^n(\Sigma_{-\ell}^m)$  can be written as a linear combination of  $\phi_{i+\frac{j}{d}}$  with ‘‘coordinates’’  $u_{i+\frac{j}{d}} := u(s_{i+\frac{j}{d}}) \in \mathbb{R}^n$ , i.e.,

$$u(s) = \left( \phi_{-\ell}(s)u_{-\ell} + \sum_{i=-\ell}^{-1} \sum_{j=1}^d \phi_{i+\frac{j}{d}}(s)u_{i+\frac{j}{d}} \right) + \sum_{i=0}^{m-1} \sum_{j=1}^d \phi_{i+\frac{j}{d}}(s)u_{i+\frac{j}{d}}.$$

Let us define  $u_{\cdot,0}$  as the vector of length  $N := n(\ell d + 1)$  that groups the  $u_{i+\frac{j}{d}}$  from the first, bracketed part of the summation above and  $u_{0\blacktriangleright 1}$  as the vector of length  $N_e := nmd$  that groups the  $u_{i+\frac{j}{d}}$  from the second part of this summation. Clearly,  $\dim \mathbb{P}_d^n(\Sigma_{-\ell}^m) = N + N_e$ . Analogous to  $u_{\cdot,0}$ , we define a vector  $u_{\cdot,1}$ , consisting of the last  $N$  components of the vector  $[u_{\cdot,0}^T \ u_{0\blacktriangleright 1}^T]^T$ . Hence,

$$u_{\cdot,0} := \begin{bmatrix} u_{-\ell} \\ \vdots \\ u_{i+\frac{j}{d}} \\ \vdots \\ u_0 \end{bmatrix}, \quad u_{\cdot,1} := \begin{bmatrix} u_{m-\ell} \\ \vdots \\ u_{i+\frac{j}{d}} \\ \vdots \\ u_m \end{bmatrix} \in \mathbb{R}^N \quad \text{and} \quad u_{0\blacktriangleright 1} := \begin{bmatrix} u_{\frac{1}{d}} \\ \vdots \\ u_{i+\frac{j}{d}} \\ \vdots \\ u_m \end{bmatrix} \in \mathbb{R}^{N_e}. \quad (10)$$

Define a set of *collocation points*  $\mathcal{C}(\Sigma^m) := \bigcup_{i=0}^{m-1} \bigcup_{\nu=1}^d \{c_{i,\nu} := s_i + c_\nu \Delta s_i\}$  on  $[0, 1]$  based on given *collocation parameters*  $c_1 < c_2 < \dots < c_d$  on  $[0, 1]$  with  $c_1 \neq 0$  or  $c_d \neq 1$ . As in AUTO and DDE-BIFTOOL, we use the *Gauss points*, i.e., the zeros of the *Legendre polynomial* of degree  $d$  on  $[0, 1]$ . Note that this choice improves the accuracy of the orbit by one order compared to arbitrary collocation points, i.e.,  $\max_{s \in [0,1]} |y(s) - u(s)| = \mathcal{O}(h^{d+1})$  instead of  $\mathcal{O}(h^d)$ , with  $h := \max_{i=0, \dots, m-1} \Delta s_i$  ([8, 11, 9]). However, for DDEs it does not lead to superconvergence (i.e.,  $\mathcal{O}(h^{2d})$  accuracy) at the mesh points  $s_i$ , contrary to the ODE case.

Our approximation to  $(y^*, T^*)$  is  $(u, T)$ , where  $u \in \mathbb{P}_d^n(\Sigma_{-\ell}^m)$  satisfies the DDE system (5a) in the finite set of points  $\mathcal{C}(\Sigma^m)$  and also satisfies certain discrete counterparts of (5b) and (5c). Our discretization of the periodicity condition (5b) requires the equality of  $u_{\cdot,0}$  and  $u_{\cdot,1}$ , or, equivalently, the equality of the restriction of  $u(s)$  to the intervals  $[s_{-\ell}, 0] = [s_{-\ell}, s_0]$  and  $[s_{m-\ell}, 1] = [s_{m-\ell}, s_m]$ , respectively. Note that the length of these intervals, namely  $-s_{-\ell}$ , exceeds  $\tau/T$  if the point  $1 - \tau/T$  does not belong to the mesh  $\Sigma^m$ . For  $u \in \mathbb{P}_d^n(\Sigma_{-\ell}^m)$  and reference solution  $u^\circ \in \mathbb{P}_d^n(\Sigma_{-\ell}^m)$ , phase condition (6) is integrated exactly by using a Gauss quadrature rule of degree  $d$  on each mesh interval  $[s_i, s_{i+1}]$ . Hence, the discretization of (5) reads

$$r_1(u_{\cdot,0}, u_{0\blacktriangleright 1}, T) = 0_{N_e \times 1}, \quad (11a)$$

$$r_2(u_{\cdot,0}, u_{\cdot,1}) := u_{\cdot,1} - u_{\cdot,0} = 0_{N \times 1}, \quad (11b)$$

$$\alpha(u_0, u_{0\blacktriangleright 1}) = 0, \quad (11c)$$

where (11a) groups the  $md$  vector-valued collocation requirements

$$\mathcal{F}(u, T)(c_{i,\nu}) - u'(c_{i,\nu}) = \mathbf{0}_{n \times 1}, \quad (12)$$

for  $i = 0, \dots, m-1$  and  $\nu = 1, \dots, d$ . Obviously, our choice of basis functions  $\phi_{i+\frac{j}{d}}$ , allows to write the restriction  $u|_{[s_i, s_{i+1}]}$ , for  $i = -\ell, \dots, m-1$ , in the form

$$u|_{[s_i, s_{i+1}]}(s) = \sum_{j=0}^d \phi_{i+\frac{j}{d}}(s) u_{i+\frac{j}{d}} = \sum_{j=0}^d \psi_j \left( \frac{s - s_i}{\Delta s_i} \right) u_{i+\frac{j}{d}}, \quad (13)$$

where

$$\psi_j(\xi) := \prod_{k=0, k \neq j}^d \frac{\xi d - k}{j - k}, \quad \text{for } j = 0, \dots, d,$$

are the *Lagrange polynomials* of degree  $d$  on  $[0, 1]$ . Hence,  $\psi_j(\hat{j}/d) = \delta_{j,\hat{j}}$  for  $j, \hat{j} = 0, \dots, d$ . Substitution of (4) and (13) in the collocation requirements (12) gives

$$Tf(\vec{u}_{i,\nu}) - \sum_{j=0}^d \frac{\psi'_j(c_\nu)}{\Delta s_i} u_{i+\frac{j}{d}} = \mathbf{0}_{n \times 1}, \quad (14)$$

with  $\vec{u}_{i,\nu} := (u(c_{i,\nu}), \dots, u(c_{i,\nu} - \tau_\kappa/T))$ , for  $i = 0, \dots, m-1$  and  $\nu = 1, \dots, d$ . In order to evaluate  $u(c_{i,\nu} - \tau_\kappa/T)$ , we use (13) with  $u$  restricted to the mesh interval  $[s_{\hat{i}_{i,\nu,k}}, s_{\hat{i}_{i,\nu,k}+1}]$ , where index  $\hat{i}_{i,\nu,k}$  is chosen from the set  $\{-\ell, \dots, m-1\}$  such that  $s_{\hat{i}_{i,\nu,k}} \leq c_{i,\nu} - \tau_\kappa/T < s_{\hat{i}_{i,\nu,k}+1}$ . Remark that  $\hat{i}_{i,\nu,0} \equiv i$ , since  $\tau_0 = 0$ .

### 3.2 Linearization.

The nonlinear collocation system (11) consists of  $N + N_e + 1$  equations and the same number of unknowns. A typical structure for the linearized system in the case of one delay (i.e.,  $\kappa = 1$ ) is depicted in Fig. 1. Let us first specify its entries. The first  $N_e$  rows are the linearization of (11a), namely,

$$\begin{aligned} & \sum_{j=0}^d \left( T \left( \sum_{k=0}^{\kappa} \psi_j \left( \frac{(c_{i,\nu} - \tau_k/T) - s_{\hat{i}_{i,\nu,k}}}{\Delta s_{\hat{i}_{i,\nu,k}}} \right) \frac{\partial f}{\partial y^k}(\cdot) \Delta u_{\hat{i}_{i,\nu,k} + \frac{j}{d}} \right) - \frac{\psi'_j(c_\nu)}{\Delta s_i} \Delta u_{i+\frac{j}{d}} \right) \\ & + \left( f(\cdot) + \frac{1}{T} \sum_{k=1}^{\kappa} \frac{\tau_k}{\Delta s_{\hat{i}_{i,\nu,k}}} \frac{\partial f}{\partial y^k}(\cdot) \sum_{j=0}^d \psi'_j \left( \frac{(c_{i,\nu} - \tau_k/T) - s_{\hat{i}_{i,\nu,k}}}{\Delta s_{\hat{i}_{i,\nu,k}}} \right) u_{\hat{i}_{i,\nu,k} + \frac{j}{d}} \right) \Delta T \\ & = - (Tf(\cdot) - u'(c_{i,\nu})), \quad \text{for } i = 0, \dots, m-1, \quad \nu = 1, \dots, d, \end{aligned} \quad (15)$$

where  $f(\cdot)$  and its partial derivatives are evaluated at  $(y^0, \dots, y^\kappa) = \vec{u}_{i,\nu}$ . Using (10) and

$$A := \frac{\partial r_1}{\partial u_{\cdot,0}} \in \mathbb{R}^{N_e \times N}, \quad B := \frac{\partial r_1}{\partial u_{0 \blacktriangleright 1}} \in \mathbb{R}^{N_e \times N_e} \quad \text{and} \quad r_{1,T} := \frac{\partial r_1}{\partial T} \in \mathbb{R}^{N_e}, \quad (16)$$

the terms in (15) can be reordered to obtain

$$A \Delta u_{\cdot,0} + B \Delta u_{0 \blacktriangleright 1} + \Delta T r_{1,T} = -r_1. \quad (17a)$$

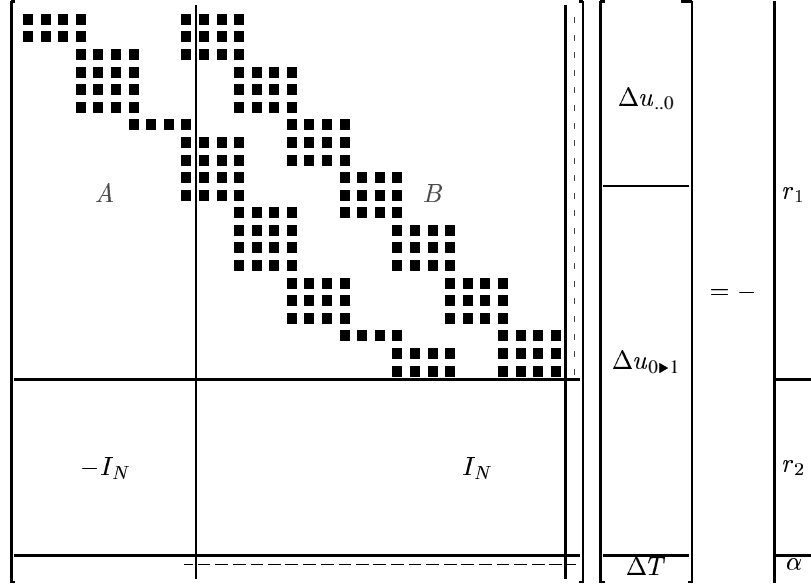


Figure 1: The structure of the linearized collocation system for one delay ( $\kappa = 1$ ) that is smaller than the (approximate) period  $T$ , using a non-equidistant mesh with  $m = 7$  mesh intervals and piecewise polynomials of degree  $d = 3$ . Here, the extended mesh  $\Sigma_{-\ell}^m$  contains  $\ell = 3$  additional mesh intervals. (“■” represents a  $n \times n$  block.)

The second block row, consisting of  $N$  rows,

$$\Delta u_{..1} - \Delta u_{..0} = -r_2, \quad (17b)$$

comes from the periodicity condition (11b). If the maximal delay  $\tau$  is larger than the (approximate) period  $T$ , the two  $N \times N$  identity matrices in this block row overlap.

The last row of the linearized system is obtained from the phase condition (11c),

$$[0_{1 \times (N-n)} \quad \alpha_{u_0}] \Delta u_{..0} + \alpha_{u_{0>1}} \Delta u_{0>1} + \alpha_T \Delta T = -\alpha, \quad (17c)$$

where  $\alpha_{u_0}$ ,  $\alpha_{u_{0>1}}$  and  $\alpha_T$  are partial derivatives. Remark that  $\alpha_T \equiv 0$  for phase condition (6).

Let us now return to the Jacobian matrix of size  $N + N_e + 1$  in Fig. 1. If there were no delays (i.e.,  $\kappa = 0$ ) then  $N = n$  and the main block, of size  $N_e \times (N + N_e)$ , of the Jacobian matrix would only consist of the rightmost, regularly shaped, staircase band of  $m$  blocks of size  $nd \times n(d + 1)$  each. AUTO uses this linearized collocation system when computing periodic solutions of ODEs ([6, Fig. 1.3]). Each delay  $\tau_k$  gives rise to an extra band. The shape of those bands is irregular and different from the rightmost band, unless an equidistant mesh is used, i.e.,  $\Delta s_i \equiv 1/m$ .

DDE-BIFTOOL ([11, 13, 12]) uses a slight variant of this scheme. The unknown orbit  $y$  is discretized using the mesh  $\Sigma^m$  on  $[0, 1]$  and  $\vec{u}_{i,\nu}$  in (14) is replaced with

$$\vec{u}_{i,\nu} := (u(c_{i,\nu}), u((c_{i,\nu} - \tau_1/T) \bmod 1), \dots, u((c_{i,\nu} - \tau_\kappa/T) \bmod 1)).$$

The resulting linear system is of size  $n + N_e + 1$  ([11, Fig. 1]). If  $u(s)$  is 1-periodic, it can also be obtained by eliminating the  $N - n = n\ell d$  columns of the matrix in Fig. 1 corresponding to the

unknowns  $u_{-\ell}, u_{-\ell+1/d}, \dots, u_{-1/d}$  using the corresponding rows of the second block row, i.e., (17b). The extra bands in the main block, corresponding to the delays, are “folded” and become circular bands ([12, Fig. 2]). DDE-BIFTOOL uses the Matlab solver for dense matrices. On a fine mesh, this approach is expensive in terms of computation time and memory requirements. The structure is hard to exploit in a direct method. As we shall see in the following sections, our collocation variant leads to the development of an efficient iterative linear system solver.

### 3.3 Discretization and linearization commute.

In this section, we show that the linearized collocation system (17) is also obtained by discretizing the linearization of the continuous BVP (5) about  $u(s)$ . This strategy is called *quasi-linearization* in [1].

The linearization of the BVP (5), for given  $(u, T)$ , is the linear BVP in the unknowns  $(\Delta y, \Delta T)$  given by

$$(\Delta y)'(s) = \mathcal{F}_y(u, T)\Delta y(s) + \Delta T \mathcal{F}_T(u, T)(s) + \mathbf{r}_1(s), \quad \text{for } s \in [0, 1], \quad (18a)$$

$$\Delta y_1(\vartheta) - \Delta y_0(\vartheta) = u|_{[s-\ell, 0]}(\vartheta) - u|_{[s_m-\ell, 1]}(\vartheta + 1), \quad \text{for } \vartheta \in [s-\ell, 0], \quad (18b)$$

$$\alpha_y \Delta y + \alpha_T \Delta T = -\alpha(u), \quad \text{on } [0, 1], \quad (18c)$$

with Fréchet derivatives  $\mathcal{F}_y$  and  $\mathcal{F}_T$  and abbreviation

$$\mathbf{r}_1(s) := \mathcal{F}(u, T)(s) - u'(s).$$

Note that the linearized DDE system (18a) extends the variational equations (8).

We now observe

**Proposition 3.1.** *The discretization of the linearized BVP (18) using the Runge-Kutta collocation method from section 3.1 results in the linearized system (17).*

*Proof.* We denote the piecewise polynomial in  $\mathbb{P}_d^n(\Sigma_{-\ell}^m)$  that approximates  $\Delta y(s)$  by  $\Delta u(s)$ . Then, vectors  $\Delta u_{\cdot, 0}$ ,  $\Delta u_{0, \bullet 1}$  and  $\Delta u_{\cdot, 1}$  can be defined, analogous to (10). We now show that they correspond to the unknowns of the linear system (17). First, replacing  $u$  in the collocation requirements (12) with  $\Delta u$  and replacing  $\mathcal{F}(u, T)(s)$  with the right hand side of the linear DDE system (18a) gives (15), or, after reordering, (17a). Finally, (17b) and (17c) are clearly discretizations of (18b) and (18c), respectively.  $\square$

This shows that the nonlinear BVP (5) can also be solved using Newton iterations for the continuous problem, linearizing about an approximate  $u(s)$  for the orbit taken from a space of piecewise polynomials. When the linearized BVP (18) at each Newton iteration is solved using a collocation scheme with the same function space as for  $u(s)$ , one obtains exactly the same discrete problem as one obtains when the collocation scheme is used to discretize the nonlinear BVP and the resulting discrete system is linearized.

Hence, due to the commutation of time integration and linearization for Runge-Kutta collocation, (17) can be obtained in two ways. Remark that Proposition 3.1 can be extended to all classical collocation, linear multistep, Runge-Kutta and finite difference schemes. In fact, it holds for all discretization schemes that approximate  $y$  by a certain  $u$  by using requirements that are linear in  $u(\cdot)$  and  $\mathcal{F}(u, T)(\cdot)$ .

### 3.4 The monodromy matrix.

From comparing the variational equations (8) with (18a), it is clear that solving

$$A\Delta u_{..0} + B\Delta u_{0\blacktriangleright 1} = 0_{N_e \times 1}$$

(i.e., (17a) with  $\Delta T = 0$  and  $r_1 = 0_{N_e \times 1}$ ) for  $\Delta u_{0\blacktriangleright 1}$ , given  $\Delta u_{..0}$  is a numerical time integrator for (8). Indeed, the orbit on  $]0, 1]$  is computed, given the initial condition  $\Delta u_{..0}$  on  $[s_{-\ell}, 0] \supseteq [-\tau/T, 0]$ . This is also equivalent to computing  $\Delta u_{0\blacktriangleright 1} = M_e \Delta u_{..0}$  with  $M_e := -B^{-1}A$ . Here,  $M_e$  is clearly the discrete equivalent of the operator  $\mathcal{M}_e(y_0)$ . Let us now define the matrix  $\tilde{I}_{K,L} := \begin{bmatrix} 0_{K \times (L-K)} & I_K \end{bmatrix}$ , i.e.,  $\tilde{I}_{K,L}v$  consists of the last  $K$  components of  $v \in \mathbb{R}^L$ . Obviously, (10) implies that

$$\Delta u_{..1} = \tilde{I}_{N,N_e} \Delta u_{0\blacktriangleright 1}, \quad \text{if } N \leq N_e, \text{ and} \quad (19)$$

$$\Delta u_{..1} = \begin{bmatrix} \tilde{I}_{N-N_e,N} \Delta u_{..0} \\ \Delta u_{0\blacktriangleright 1} \end{bmatrix}, \quad \text{otherwise.} \quad (20)$$

Consider first the case  $N \leq N_e$ . Then  $\Delta u_{..1} = M \Delta u_{..0}$  with  $M := \tilde{I}_{N,N_e} M_e$  is the final state (i.e., defined on  $[s_{m-\ell}, 1] \supseteq [1 - \tau/T, 1]$ ) of numerical time integration of (8) with the discretized initial condition  $\Delta u_{..0}$ . Hence,  $M$  is the discrete equivalent of the monodromy operator  $\mathcal{M}(y_0)$  and is therefore called the *monodromy matrix*. If  $N > N_e$ , according to (20), the final state is given by  $\Delta u_{..1} = M \Delta u_{..0}$  with

$$M := \begin{bmatrix} \tilde{I}_{N-N_e,N} \\ M_e \end{bmatrix}.$$

The results are summarized in

**Definition 3.1.** *The matrix*

$$M_e := -B^{-1}A \in \mathbb{R}^{N_e \times N}, \quad (21)$$

with  $A$  and  $B$  from (16) is the discretization of the operator  $\mathcal{M}_e(y_0)$ . Likewise, the monodromy matrix

$$M := \tilde{I}_{N,N_e} M_e, \quad \text{if } N \leq N_e, \text{ and} \quad (22)$$

$$M := \begin{bmatrix} \tilde{I}_{N-N_e,N} \\ M_e \end{bmatrix} \quad \text{if } N > N_e. \quad (23)$$

is the discretization of the monodromy operator  $\mathcal{M}(y_0)$ .

The dominant eigenvalues of  $M$  are usually good approximations of the dominant eigenvalues of  $\mathcal{M}$ . Therefore, the eigenvalues of  $M$  are also called “the Floquet multipliers”. The smaller eigenvalues of  $M$  are usually not very good approximations of eigenvalues of  $\mathcal{M}$ . However, only the dominant Floquet multipliers are important in determining the stability of the periodic solution.

## 4 Solving the linearized BVP with single shooting.

In section 4.1 we motivate the use of single shooting for solving the linearized BVP (18). The procedure is discussed in section 4.2. In section 4.3, we will show how matrix-vector products with the monodromy matrix can be computed efficiently.

## 4.1 Motivation.

Let us first briefly turn to the ODE case for an analogy. According to section 3.2, the linearized collocation system in AUTO is a special case of (17) with  $N = n$  and with only the rightmost band. AUTO's linear system solver fully exploits this structure ([6]). In the first step, the unknowns inside each mesh interval  $[s_i, s_{i+1}]$ , i.e.,  $\Delta u_{i+\frac{j}{d}}$  for  $j = 1, \dots, d-1$ , are eliminated using Gauss elimination with row pivoting within each  $nd \times n(d+1)$  subblock. Note that this can be done in parallel for each mesh interval. In the second step, the unknowns  $\Delta u_2, \dots, \Delta u_{m-1}$  are eliminated one by one. The result is an (underdetermined) linear system in  $\Delta u_0$ ,  $\Delta u_m$  and  $\Delta T$ . Solving  $\Delta u_m$  from this system given  $\Delta u_0$  and  $\Delta T$  is numerically equivalent to integration of (18a) with the Gauss-Legendre Runge-Kutta scheme. In the third step, AUTO computes  $\Delta u_0$ ,  $\Delta u_m$  and  $\Delta T$  from the condensed linearized collocation system augmented with the periodicity condition and the phase condition (with all unknowns except  $\Delta u_0$  and  $\Delta u_m$  also eliminated in the first two steps). Solving this linear system is equivalent to single shooting for the linearized BVP (18) (though in the resulting linear system, one usually first eliminates  $\Delta u_m$  also using the periodicity condition). Finally, the other unknowns are computed via back-substitution. Since the single shooting is only performed at the linearized level, one of the main problems of shooting methods is avoided, namely the small domain of attraction of the Newton iteration. This motivates the development of a similar single shooting-like procedure for our collocation variant for DDEs.

## 4.2 Condensation of the linearized system.

Analogously to AUTO, single shooting for the linearized BVP (18) corresponds to a condensation of the linearized system (17). Our goal now is to obtain a linear system in  $\Delta u_{..0}$  and  $\Delta T$ . Reordering (17a), premultiplying with  $B^{-1}$  and using (21), one obtains

$$\Delta u_{0\triangleright 1} = M_e \Delta u_{..0} + \Delta T (-B^{-1} r_{1,T}) + (-B^{-1} r_1). \quad (24)$$

Computing  $\Delta u_{0\triangleright 1}$  from this system, given  $\Delta u_{..0}$  and  $\Delta T$ , is equivalent to numerical time integration of (18a) over the interval  $]0, 1]$  with initial condition  $\Delta u_{..0}$ . Integrating (18a) numerically with the initial condition  $\Delta u_{..0} = 0_{N \times 1}$  and  $\Delta T = 0$  computes  $-B^{-1} r_1$ .

To extract  $\Delta u_{..1}$  from (24), we must distinguish between two cases. If  $N \leq N_e$ , we use (19) and select the last  $N$  rows of (24) by premultiplying with  $\tilde{I}_{N, N_e}$  to obtain  $\Delta u_{..1}$ . Defining

$$z := \tilde{I}_{N, N_e} (-B^{-1} r_1) \quad \text{and} \quad z_T := \tilde{I}_{N, N_e} (-B^{-1} r_{1,T})$$

and using (22), we obtain

$$\Delta u_{..1} = M \Delta u_{..0} + \Delta T z_T + z. \quad (25)$$

Remark that  $z_T$  is not the partial derivative of  $z$  w.r.t. the (approximate) period  $T$ , since  $B$  depends on  $T$  also.

If, on the other hand,  $N > N_e$ , we use (20) and obtain  $\Delta u_{..1}$  by augmenting  $\tilde{I}_{N-N_e, N} \Delta u_{..0}$  with the right hand side of (24). Using (23) and defining

$$z := \begin{bmatrix} 0_{(N-N_e) \times 1} \\ -B^{-1} r_1 \end{bmatrix} \quad \text{and} \quad z_T := \begin{bmatrix} 0_{(N-N_e) \times 1} \\ -B^{-1} r_{1,T} \end{bmatrix}$$

we obtain again (25).

Equation (25) is the cornerstone of our linear single shooting procedure. Indeed, this equation, obtained from the linearized collocation system, also corresponds to integration of the linearized DDE system (18a) over  $\Delta s = 1$  using a sequence of implicit Runge-Kutta steps with step sizes  $\Delta s_i$ .

Using the periodicity condition (17b), we eliminate  $\Delta u_{..1}$  from (25) to obtain

$$(M - I_N)\Delta u_{..0} + \Delta T z_T = -r_c \quad (26)$$

with

$$r_c := z + r_2. \quad (27)$$

Substituting (24) in the linearized phase condition (17c), one obtains

$$a_u^T \Delta u_{..0} + \Delta T a_T = -\alpha_c, \quad (28)$$

with

$$a_u^T := [0_{1 \times (N-n)} \ \alpha_{u_0}] + \alpha_{u_{0 \triangleright 1}} M_e, \quad a_T := \alpha_T + \alpha_{u_{0 \triangleright 1}} (-B^{-1} r_{1,T}) \quad \text{and} \\ \alpha_c := \alpha + \alpha_{u_{0 \triangleright 1}} (-B^{-1} r_1).$$

Combining (26) with (28) results in the linear single shooting system

$$\begin{bmatrix} M - I_N & z_T \\ a_u^T & a_T \end{bmatrix} \begin{bmatrix} \Delta u_{..0} \\ \Delta T \end{bmatrix} = - \begin{bmatrix} r_c \\ \alpha_c \end{bmatrix}. \quad (29)$$

This condensed system consists of a  $N \times N$ -block with one border column at the right and one border row at the bottom.

Though the structure is identical, (29) is not the linearized single shooting system for the nonlinear BVP (5). Indeed,  $r_c$  is not the discretization of  $\mathcal{T}(y_0, 1) - y_0$ , with  $\mathcal{T}$  the time integration operator defined in section 2.2, and  $z_T$  is not the partial derivative of  $r_c$  w.r.t.  $T$ .

It is clear that solving (29) and then computing  $\Delta u_{0 \triangleright 1}$  from (24) is equivalent (in exact arithmetic) to solving (17). Hence, the above procedure (solving the linearized BVP (18) repeatedly using the above single shooting procedure) is fully equivalent to using Newton iterations with a direct solver for the linearized collocation system.

However, it is expensive to use a direct solver for the linear system (17) or the condensed system (29). Instead, in section 5, we will derive an iterative solver to compute an approximate solution to (29) and analyze the resulting algorithm.

### 4.3 Computing the Floquet multipliers.

The monodromy matrix  $M$  is easily computed from its definition. However, this matrix requires a lot of storage, especially when the maximal delay  $\tau$  is large compared to the (approximate) period  $T$ , if a fine discretization is used or if the DDE system is the space discretization of a delay PDE. However, we do not need to compute  $M$  explicitly to compute the matrix-vector product  $Mv$ . According to Definition 3.1,  $Mv$  can be computed by first solving the system  $Bw = -Av$  for  $w$  using a block version of forward substitution. Afterwards,  $Mv$  is easily obtained using

$$Mv = \tilde{I}_{N, N_e} w, \quad \text{if } N \leq N_e, \text{ and} \\ Mv = \begin{bmatrix} \tilde{I}_{N-N_e, N} v \\ w \end{bmatrix} \quad \text{if } N > N_e.$$

This algorithm is efficient since it only requires the solution of  $m$  square systems of size  $nd \times nd$ . The LU factors of the diagonal blocks of  $B$  can be saved if the matrix-vector products have to be

recomputed. This algorithm to compute the matrix-vector product with the monodromy matrix enables the use of orthogonal subspace iteration or Krylov methods ([2]) to obtain the dominant Floquet multipliers. Moreover, it is the key element in the construction of an iterative solver for (29).

## 5 The Newton-Picard method.

In each Newton iteration, we first solve a condensed linear system of the form (29). Afterwards, the other unknowns in (17) are computed using (24). System (29) has the same structure as the linearized single shooting system in [18] that is solved with the Newton-Picard method. The Newton-Picard method can be used to solve (29) because the monodromy matrix  $M$  has typically only few dominant eigenvalues and because matrix-vector products with  $M$  can be computed efficiently. This method combines a direct and an iterative solver, but a single step will only compute an approximate solution to (29). However, contrary to previous work, we will use more than one Newton-Picard step at each Newton iteration to solve (29) with a higher accuracy. Otherwise, the overall algorithm would behave more like a single shooting method for the nonlinear BVP with a Runge-Kutta time integrator rather than like a collocation method. Indeed,  $u_{0\triangleright 1}$  is computed exactly from (24) since we use Gauss elimination to solve the system with the block lower triangular matrix  $B$ . However, because (29) is not solved accurately, the periodic boundary condition (17b) is not satisfied. As a result,  $[u_{\cdot,0}^T \ u_{0\triangleright 1}^T]^T$  will converge quadratically to an orbit of the DDE system, while the convergence to a periodic (i.e., closed) orbit will only be linear (the convergence rate of the Newton-Picard method).

### 5.1 The basic Newton-Picard step.

Consider the linear system

$$\begin{bmatrix} M - I & z_T \\ a_u^T & a_T \end{bmatrix} \begin{bmatrix} \delta u_{\cdot,0} \\ \delta T \end{bmatrix} = - \begin{bmatrix} r \\ \alpha \end{bmatrix} \quad (30)$$

with an arbitrary right hand side  $r$  and  $\alpha$ . Solving this system with a direct method is expensive because of the large size and the high cost of computing  $M$ . On the other hand, given  $\delta T$ , one could use the Picard iteration

$$\delta u_{\cdot,0}^{[i+1]} = M \delta u_{\cdot,0}^{[i]} + \delta T z_T + r \quad (31)$$

to compute  $\delta u_{\cdot,0}$ . However, this iteration only converges if all eigenvalues of  $M$  lie within the unit circle, and convergence is slow if an eigenvalue is close to the unit circle. Moreover, this iteration assumes that we know  $\delta T$ . The Newton-Picard method combines the best of both methods. It isolates the few modes of  $M$  responsible for slow convergence or divergence of (31) and combines a direct method for a small linear system with Picard iterations using a high-dimensional projection of (31).

Let us choose a threshold  $\rho$  with  $0 < \rho < 1$ . Let  $\mathcal{V}_p$  be the maximal invariant subspace of  $\mathbb{R}^N$  (i.e., the generalized eigenspace) corresponding to all Floquet multipliers  $\mu_1, \dots, \mu_p$  with modulus larger than or equal to  $\rho$ . The threshold  $\rho$  is chosen such that this subspace includes only the unstable and a few weakly stable modes. As argued above, the dimension  $p$  of  $\mathcal{V}_p$  is small compared to  $N$ . Let the columns of  $V_p \in \mathbb{R}^{N \times p}$  be an orthonormal basis for  $\mathcal{V}_p$ . Furthermore, let  $V_q \in \mathbb{R}^{N \times (N-p)} = \mathbb{R}^{N \times q}$  be an orthonormal basis for the orthogonal complement  $\mathcal{V}_p^\perp$  of  $\mathcal{V}_p$ . Since  $M$  is in general a nonnormal matrix,  $\mathcal{V}_p^\perp$  is not an invariant subspace of  $M$ . It is clear that computing  $V_q$  is too expensive and we will eliminate the use of this basis from the final algorithm. We can construct orthogonal projectors  $P$  and  $Q$  of  $\mathbb{R}^N$  onto  $\mathcal{V}_p$  and  $\mathcal{V}_p^\perp$ , respectively, as

$$P := V_p V_p^T, \quad \text{and} \quad Q := V_q V_q^T = I_N - V_p V_p^T.$$

Note that we do not need the basis  $V_q$  to apply the projector  $Q$  to a vector. For any  $\delta u_{..0} \in \mathbb{R}^N$ , there is the unique decomposition

$$\delta u_{..0} = V_p \delta \bar{p} + V_q \delta \bar{q} \quad \text{with} \quad \delta p := V_p \delta \bar{p} := P \delta u_{..0}, \quad \delta q := V_q \delta \bar{q} := Q \delta u_{..0} \quad (32)$$

where  $\delta \bar{p} \in \mathbb{R}^p$  and  $\delta \bar{q} \in \mathbb{R}^{N-p}$ . Substituting (32) in (30) and premultiplying the first  $N$  equations with  $[V_q \ V_p]^T$ , one obtains

$$\begin{bmatrix} V_q^T M V_q - I_q & 0_{q \times p} & V_q^T z_T \\ V_p^T M V_q & V_p^T M V_p - I_p & V_p^T z_T \\ a_u^T V_q & a_u^T V_p & a_T \end{bmatrix} \begin{bmatrix} \delta \bar{q} \\ \delta \bar{p} \\ \delta T \end{bmatrix} = - \begin{bmatrix} V_q^T r \\ V_p^T r \\ \alpha \end{bmatrix} \quad (33)$$

where we used  $V_q^T V_p = 0_{q \times p}$ ,  $V_p^T V_q = 0_{p \times q}$  and  $V_q^T M V_p = 0_{q \times p}$ . The latter equality holds since  $\mathcal{V}_p$  is an invariant subspace of  $\mathbb{R}^N$ . In actual computations,  $V_p$  will only be an approximate basis for that subspace, but we will put the term to zero anyway. It is easy to show that the eigenvalues of  $V_q^T M V_q$  are the eigenvalues  $\mu_{p+1}, \dots, \mu_N$  of  $M$  ([22]). These eigenvalues are smaller than 1 in modulus by definition of the subspace  $\mathcal{V}_p$  and hence the matrix  $V_q^T M V_q - I_q$  is nonsingular. To solve (33), we first reduce the system to a block upper triangular system by eliminating the (2,1) and (3,1) blocks using the (1,1) block. The lower  $p+1$  equations of the resulting system are

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T (z_T + M V_q \delta \bar{q}_T) \\ a_u^T V_p & a_T + a_u^T V_q \delta \bar{q}_T \end{bmatrix} \begin{bmatrix} \delta \bar{p} \\ \delta T \end{bmatrix} = - \begin{bmatrix} V_p^T (r + M V_q \delta \bar{q}_r) \\ \alpha + a_u^T V_q \delta \bar{q}_r \end{bmatrix}, \quad (34)$$

with

$$\delta \bar{q}_r = - (V_q^T M V_q - I_q)^{-1} V_q^T r \quad \text{and} \quad \delta \bar{q}_T = - (V_q^T M V_q - I_q)^{-1} V_q^T z_T, \quad (35)$$

i.e.,  $\delta \bar{q}_r$  and  $\delta \bar{q}_T$  are the solution of the linear systems

$$(V_q^T M V_q - I_q) \delta \bar{q}_r = -V_q^T r \quad \text{and} \quad (V_q^T M V_q - I_q) \delta \bar{q}_T = -V_q^T z_T. \quad (36)$$

The matrix in (34) is of full rank *iff* the matrix in (30) is of full rank. Since (34) is only a small system, it can be solved easily using Gaussian elimination. Next, we compute

$$\delta \bar{q} = - (V_q^T M V_q - I_q)^{-1} V_q^T (r + \delta T z_T) = \delta \bar{q}_r + \delta T \delta \bar{q}_T, \quad (37)$$

which requires only already computed information. Finally,  $\delta u_{..0}$  is obtained from (32).

There are two problems with the above procedure. First, we need to determine the basis  $V_p$ . Furthermore, it is prohibitively expensive to build the matrix  $V_q^T M V_q$  and to compute  $\delta \bar{q}_r$  and  $\delta \bar{q}_T$  using direct methods. Let us deal with the latter first. Equation (35) can be rewritten as

$$\delta \bar{q}_* = V_q^T M V_q \delta \bar{q}_* + V_q^T b, \quad (38)$$

with  $\delta \bar{q}_*$  either  $\delta \bar{q}_r$  or  $\delta \bar{q}_T$  and  $b$  either  $r$  or  $z_T$ . Since the spectral radius  $r_\sigma(V_q^T M V_q) = |\mu_{p+1}| < \rho < 1$ , the linear Picard iteration

$$\delta \bar{q}_*^{[i+1]} = V_q^T M V_q \delta \bar{q}_*^{[i]} + V_q^T b, \quad \text{with} \quad \delta \bar{q}_*^{[0]} = 0_{q \times 1}, \quad (39)$$

converges asymptotically. In (34) and (32), we only need  $\delta q_* := V_q^T \delta \bar{q}_*$ . Therefore we can premultiply (39) with  $V_q$  resulting in the iteration

$$\delta q_*^{[i+1]} = Q \left( M \delta q_*^{[i]} + b \right) \quad (40)$$

which requires only matrix-vector products with  $M$  and projections with  $Q$ , so  $V_q$  is no longer needed. Contrary to the implementation in [16], the number of Picard iterations is not fixed in advance in our present experiments. The number is adapted dynamically to obtain better performance of the Newton-Picard method (see below). Note that although asymptotic convergence for (40) is guaranteed, the iteration may initially diverge since  $M$  is in general a nonnormal matrix.

To compute the basis  $V_p$ , we employ orthogonal subspace iteration with projection and locking ([2]). This is an iterative method which only requires matrix-vector products with  $M$ . Provided the value of  $p$  is right, it converges to the maximal invariant subspace of  $M$  corresponding to the  $p$  largest (in modulus) eigenvalues. Details of our implementation, and strategies to determine  $p$ , are described in [15, 16]. We do not compute the basis  $V_p$  with full accuracy since this would be very expensive. The required accuracy for the basis depends on two elements in the Newton-Picard method. First, for (asymptotic) convergence of (40),  $r_\sigma(V_q^T M V_q) < 1$  is required. A robust test for this is expensive since it essentially requires to compute the largest eigenvalue of  $V_q^T M V_q$  with enough accuracy to be sure that it is smaller than  $\rho$  in modulus. We do not do this, but we do use the same strategy as in [15, 16]. We keep more than  $p$  vectors in the basis for the subspace iteration in order to estimate  $\mu_{p+1}$ . This estimate is not only used to determine the correct dimension of  $\mathcal{V}_p$ , but requiring that  $|\mu_{p+1}| < \rho$  also gives some confidence that (40) will converge. Second, in deriving (33), we also used  $V_q^T M V_p = 0_{q \times p}$ . This is no longer satisfied if a basis for an approximate invariant subspace is used, but we neglect the term anyway. During the subspace iteration, we monitor

$$\|V_q^T M V_p\|_2 = \|Q M V_p\|_2 = \|M V_p - V_p (V_p^T M V_p)\|_2 \quad (41)$$

and declare convergence when this quantity has become small enough. Computing (41) is cheap since both  $M V_p$  and the projected matrix  $V_p^T M V_p$  are already computed in the algorithm for subspace iteration with projection.

## 5.2 Improving the convergence.

Since  $V_p$ ,  $\delta q_r$  and  $\delta q_T$  are approximations, one Newton-Picard step only gives an approximate solution  $(\delta u_{..0}, \delta T)$  to (30). Using this approximation would limit the convergence of the outer Newton iteration to the linear convergence of the Newton-Picard method. We will regain quadratic convergence by improving the accuracy of the solution using multiple Newton-Picard steps.

Assume that we have an approximate solution  $(\Delta u_{..0}^{(\nu)}, \Delta T^{(\nu)})$  to (29). The error  $(\delta u_{..0}^{(\nu)}, \delta T^{(\nu)}) := (\Delta u_{..0} - \Delta u_{..0}^{(\nu)}, \Delta T - \Delta T^{(\nu)})$  is a solution of the *residual equation*

$$\begin{aligned} \begin{bmatrix} M - I & z_T \\ a_u^T & a_T \end{bmatrix} \begin{bmatrix} \delta u_{..0}^{(\nu)} \\ \delta T^{(\nu)} \end{bmatrix} &= - \begin{bmatrix} r \\ \alpha \end{bmatrix} - \begin{bmatrix} M - I & z_T \\ a_u^T & a_T \end{bmatrix} \begin{bmatrix} \Delta u_{..0}^{(\nu)} \\ \Delta T^{(\nu)} \end{bmatrix} \\ &= - \begin{bmatrix} r + (M - I)\Delta u_{..0}^{(\nu)} + \Delta T^{(\nu)} z_T \\ \alpha + a_u^T \Delta u_{..0}^{(\nu)} + \Delta T^{(\nu)} a_T \end{bmatrix}. \end{aligned} \quad (42)$$

This is again a system in the form (30) which can be solved approximately by a Newton-Picard step. This observation gives rise to a loop inside each Newton iteration.

**Algorithm 5.1.** *Multiple Newton-Picard steps per Newton iteration*

Set  $\Delta u_{..0}^{(0)} = 0_{N \times 1}$ ,  $\Delta T^{(0)} = 0$  and  $\nu = 0$ ,  
Do until convergence

Compute an approximate solution  $(\delta u_{..0}^{(\nu)}, \delta T^{(\nu)})$  to (42) by a Newton-Picard step,  
 Update  $\Delta u_{..0}^{(\nu+1)} = \Delta u_{..0}^{(\nu)} + \delta u_{..0}^{(\nu)}$ ,  $\Delta T^{(\nu+1)} = \Delta T^{(\nu)} + \delta T^{(\nu)}$ ,  $\nu \leftarrow \nu + 1$ .

Note that  $M$  and  $Q_{z_T}$  do not change during the steps. Therefore the basis  $V_p$  and  $\delta q_T$  need to be computed only during the first pass through the loop. In the other passes, we must only compute  $\delta q_r$  using the fixed point iteration (40) and solve a small linear system. Hence, it is relatively cheap to obtain more accuracy than a single Newton-Picard step can deliver. To guarantee quadratic convergence of the outer Newton iteration, we assure that Algorithm 5.1 sufficiently decreases the norm of  $r + (M - I)\Delta u_{..0}^{(\nu)} + \Delta T^{(\nu)} z_T$ , see section 5.4.

### 5.3 The algorithm.

The resulting algorithm has three levels of nested loops. At the outer level there is a Newton iteration to solve the nonlinear system (11). In each Newton iteration, the large linearized system (17) (see Fig. 1) is solved approximately by solving the condensed system (29) approximately for  $\Delta u_{..0}$  and  $\Delta T$  and then computing  $\Delta u_{0\blacktriangleright 1}$  using (24). The latter is equivalent to solving (17a) exactly for  $\Delta u_{0\blacktriangleright 1}$ . Therefore, after solving the linear system, the residual of (17a) is zero except for rounding errors while the residual of (17b) and (17c) is nonzero.

The loop at the second level is outlined in Algorithm 5.1. This iteration is responsible for solving the linearized equations with enough accuracy to maintain quadratic convergence of the Newton process and to assure that the overall method behaves as a collocation method rather than as a single shooting method. In the Newton-Picard step, the basis  $V_p$  is computed iteratively using orthogonal subspace iteration and the vectors  $\delta q_*$  are computed iteratively using the Picard iteration (40), the third and innermost loop level.

#### Algorithm 5.2. Newton-Picard collocation

*Initialize the orbit, i.e., periodically extend  $u(s)$  to a function on  $[s_{-\ell}, 1]$ .*

*Initialize the basis  $V_p$ .*

*Do until convergence (outer loop level: Newton iterations)*

*Construct the condensed linearized system, i.e., compute  $z_T$ ,  $a_u$ ,  $a_T$ ,  $r_c$  and  $\alpha_c$  and prepare for efficiently computing matrix-vector products with  $M$ .*

*Set  $\Delta u_{..0} = 0_{N \times 1}$ ,  $\Delta T = 0$ .*

*Do until convergence (second loop level: Newton-Picard steps)*

*Compute or improve the basis  $V_p$  using orthogonal subspace iteration with projection and deflation.*

*Compute  $r = r_c + (M - I)\Delta u_{..0} + \Delta T z_T$ ,  $\alpha = \alpha_c + a_u^T \Delta u_{..0} + \Delta T a_T$ .*

*Set  $\delta q_r = 0_{N \times 1}$  and  $\delta q_T = 0_{N \times 1}$ .*

*(inner loop level: Picard iterations)*

*Iterate until convergence:  $\delta q_r \leftarrow Q(M\delta q_r + r)$ .*

*Iterate until convergence:  $\delta q_T \leftarrow Q(M\delta q_T + z_T)$ .*

*Solve*

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T (z_T + M \delta q_T) \\ a_u^T V_p & a_T + a_u^T \delta q_T \end{bmatrix} \begin{bmatrix} \delta \bar{p} \\ \delta T \end{bmatrix} = - \begin{bmatrix} V_p^T (r + M \delta q_r) \\ \alpha + a_u^T \delta q_r \end{bmatrix}.$$

*Update  $\Delta u_{..0} \leftarrow \Delta u_{..0} + V_p \delta \bar{p} + \delta q_r + \delta T \delta q_T$ ,  $\Delta T \leftarrow \Delta T + \delta T$ .*

*Update  $\Delta u_{0\blacktriangleright 1}$  using (24).*

Note that the monodromy matrix  $M$  and the vector  $z_T$  do not change inside the second loop, i.e., the Newton-Picard steps. Therefore it is possible to move the basis computation and the iteration for  $\delta q_T$  one level up. However, as will become clear in the next section, we provide the option in our implementation to adapt the convergence criteria for both the subspace iteration and the Picard iteration based on the observed convergence behavior to guarantee convergence of the second loop. For the sake of clarity, this mechanism is not shown in Algorithm 5.2.

#### 5.4 The convergence criteria.

To test for convergence of the Newton iteration, we can use any criterion used in a traditional Newton process, e.g., check the residual or use the Newton update of a given step as an estimate for the error at the beginning of that step. Our implementation tests for the relative size of the residuals  $r_1$  and  $r_2$  in (17a) and (17b) respectively.

The second loop level is given by Algorithm 5.1. Its convergence criterion is based on an estimate for the norm of the higher-order terms neglected in the Newton linearization, and iterations are performed until the norm of the residual

$$r_{\text{NP}} := r_c + (M - I)\Delta u_{..0} + \Delta T z_T = r + (M - I)\delta u_{..0} + \delta T z_T$$

of the first block row of (29) (with the sign reversed) decreases below that estimate. In doing so, we obtain quadratic convergence of the Newton iteration without doing excess work. We also stop the iterations if  $\|r_{\text{NP}}\|_2$  plus the estimate for the norm of the higher-order terms is small enough to give confidence that the Newton iterations have converged to the required level of accuracy. This avoids excess work in the last Newton iteration. The subspace iteration and (innermost) Picard iteration also require convergence criteria.

Let us elaborate first on the various stopping criteria involved with a Newton-Picard step (the innermost loop). The  $Q$  and  $P$  projections of the first  $N$  components of  $r_{\text{NP}}$  are

$$\begin{aligned} Qr_{\text{NP}} &= Qr + Q(M - I)\delta u_{..0} + \delta T Q z_T \quad \text{and} \\ Pr_{\text{NP}} &= Pr + P(M - I)\delta u_{..0} + \delta T P z_T, \end{aligned} \tag{43}$$

respectively. After substitution of (32) and (37) in both relations and premultiplying the second relation with  $V_p^T$ , one obtains

$$\begin{aligned} Qr_{\text{NP}} &= Qr + Q(M - I)(\delta q_r + \delta T \delta q_T + \delta p) + \delta T Q z_T \\ &= QM V_p \delta \bar{p} + (Qr + Q(M - I)\delta q_r) + \delta T (Q z_T + Q(M - I)\delta q_T) \end{aligned} \tag{44}$$

and

$$\begin{aligned} V_p^T r_{\text{NP}} &= V_p^T r + V_p^T (M - I)(\delta q_r + \delta T \delta q_T + \delta p) + \delta T V_p^T z_T \\ &= (V_p^T M V_p - I_p) \delta \bar{p} + \delta T V_p^T (z_T + M \delta q_T) + V_p^T (r + M \delta q_r). \end{aligned} \tag{45}$$

The right-hand side of (45) comprises the first  $p$  equations of the small system (34). Since this system is solved with a direct method, we can safely assume  $V_p^T r_{\text{NP}} \approx 0_{p \times 1}$ , hence  $r_{\text{NP}} \approx Qr_{\text{NP}}$ .

Expression (44) requires some more attention. The size of the first term in (44),  $QM V_p \delta \bar{p}$ , is directly related to the accuracy of the invariant subspace. This explains why we monitor (41) during the subspace iteration. Given an estimate for  $\|\delta \bar{p}\|_2$ , one could determine a criterion for  $\|QM V_p\|_2$  such that  $Qr_{\text{NP}}$  can decrease sufficiently. However, producing a precise estimate for  $\|\delta \bar{p}\|_2$  is hard. In

our present implementation, we simply set an upper limit for  $\|QMV_p\|_2$ , typically  $10^{-2}$  or less. Note that the computation of  $QMV_p\delta\bar{p}$  requires no new matrix-vector products since  $MV_p$  is computed already in the subspace iteration with projection.

The second and third term in the right hand side (44) include  $Qr + Q(M - I)\delta q_r$  and  $Qz_T + Q(M - I)\delta q_T$ , which are the residuals of the systems (36), premultiplied with  $V_q$ . Their computation is essentially free too since  $M\delta q_r$  and  $M\delta q_T$  are needed to build (34). It is clear that if we compute the basis with enough accuracy (so that  $\|QMV_p\|_2$  is small enough *and* that  $r_\sigma(V_q^T M V_q) < 1$ ) and if we do enough Picard iterations (40), it is always possible to assure that  $\|r_{\text{NP}}\|_2 < \|r\|_2$ . If we do not reach that goal, (44) can be used to check which terms are responsible. If  $QMV_p\delta\bar{p}$  is too large, we then generate a sharper bound for the subspace iterations based on the present value of  $\delta\bar{p}$  and the desired norm of  $Qr_{\text{NP}}$  at the end of the step. If one of the other two terms is too large, we change the convergence criterion for the corresponding Picard iteration. In all cases, the Newton-Picard step is restarted. For optimal performance, one should balance the size of  $Qr + Q(M - I)\delta q_r$  and  $\delta T(Qz_T + Q(M - I)\delta q_T)$ . Making one of those terms much smaller than the other one costs matrix-vector products with  $M$  while it does not improve the results. Therefore, it also makes no sense to make those terms much smaller than  $QMV_p\delta\bar{p}$ . The basis  $V_p$  should only be accurate enough to guarantee that the Picard iteration converge fast enough and that  $r_{\text{NP}}$  decreases in every Newton-Picard step. Using a more accurate basis will reduce the number of Newton-Picard steps needed at every Newton step, but every step will require more Picard iterations if we try to balance all three terms in the right-hand side of (45) to obtain the maximal reduction of  $r_{\text{NP}}$  possible in a single step. The overall number of Picard iterations at every Newton step will not decrease while one will need to do more work to compute the more accurate basis. Hence the total cost will increase.

## 6 Continuation.

In a continuation process, one of the bifurcation parameters,  $\gamma$ , is allowed to vary and a *parameterizing equation* is added to the BVP (5). We use the *pseudo-arclength* parameterizing equation,

$$\beta(y, T, \gamma) := \omega_y \int_0^1 (\delta_{\text{pr}} y(s))^T \Delta y(s) ds + \omega_T (\delta_{\text{pr}} T)^T \Delta T + \omega_\gamma (\delta_{\text{pr}} \gamma)^T \Delta \gamma = 0,$$

where  $(\delta_{\text{pr}} y, \delta_{\text{pr}} T, \delta_{\text{pr}} \gamma)$  is the *predictor direction* and the scalars  $\omega_y$ ,  $\omega_T$  and  $\omega_\gamma$  are positive weights ([20]).

The additional equations give rise to an extra border column and border row in (29). Analogously to [15, 16], Algorithm 5.2 can easily be extended to obtain a continuation variant of the Newton-Picard collocation method. The Newton-Picard method can take advantage of the continuation by using the final basis  $V_p$  from the previous periodic solution on the branch as the starting value for the subspace iterations in the computation of the next periodic solution.

## 7 Examples.

We illustrate the convergence and efficiency of Algorithm 5.2 and its continuation variant by computing periodic solutions of two models.

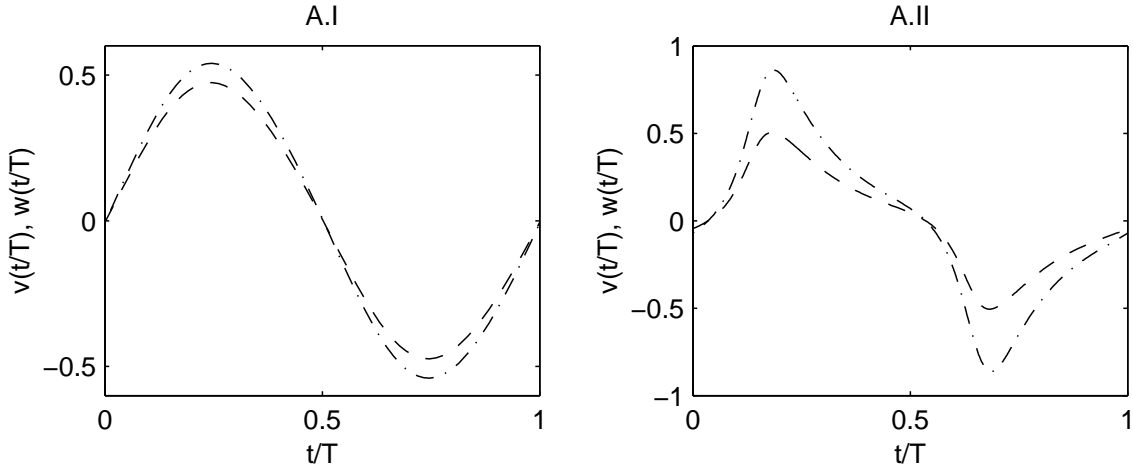


Figure 2: The orbits of the periodic solutions A.I (left) and A.II (right) with  $v$  (dashed line) and  $w$  (dash-dotted line).

**Model A** In [21], two coupled identical neurons with time-delayed connections are modeled by the system of  $n = 2$  DDEs

$$\begin{cases} v'(t) = -\lambda v(t) + \beta_0 \tanh(v(t - \tau_s)) + \beta_{1,2} \tanh(w(t - \tau_2)), \\ w'(t) = -\lambda w(t) + \beta_0 \tanh(w(t - \tau_s)) + \beta_{2,1} \tanh(v(t - \tau_1)). \end{cases}$$

We computed two periodic solutions on a branch obtained by varying  $\beta_{2,1}$  and fixing the parameters  $\lambda = 0.5$ ,  $\beta_0 = -1$ ,  $\beta_{1,2} = 1$ ,  $\tau_1 = 0.2$ ,  $\tau_2 = 0.2$  and  $\tau_s = 1.5$  ([8, 13]). The first periodic solution was computed by fixing  $\beta_{2,1} = 1.27406$  as well and using Algorithm 5.2. We obtained a stable periodic solution with period  $T \approx 10.0174$ , which we denote by A.I. The second periodic solution denoted by A.II, was obtained by using the continuation variant of the Newton-Picard collocation method with bifurcation parameter  $\beta_{2,1}$ . Periodic solution A.II has  $\beta_{2,1} \approx 2.35001$  and  $T \approx 66.3164$ . Note that A.II is unstable because the branch of periodic solutions has passed through a *limit point of cycles*. The period of A.II is large because the branch approaches a heteroclinic orbit. The orbits of A.I and A.II are depicted in Fig. 2.

**Model B** In [3], the mammalian platelet production is modeled by the scalar DDE

$$x'(t) = -\gamma x(t) + g(x(t - \tau_m)) - g(x(t - \tau_m - \tau_s))e^{-\gamma\tau_s}$$

with  $g(x) := g_0 \zeta^\lambda x / (\zeta^\lambda + x^\lambda)$ . Like [17], we fixed  $\gamma = 12$ ,  $g_0 = 27000$ ,  $\zeta = 0.04$ ,  $\tau_m = 9$  and  $\tau_s = 10$  and varied  $\lambda$ . Using the continuation variant of our method, we computed a stable periodic solution with period  $T \approx 18.20$  at  $\lambda \approx 2.135$ , further denoted by B. The orbit is shown in Fig. 3 (left).

The periodic solutions were computed with a tolerance of  $10^{-10}$  for the relative size of the residuals  $r_1$  and  $r_2$ . We used a basis threshold  $\rho = 0.5$ , piecewise polynomials of degree  $d = 3$  and the same mesh adaption strategy as DDE-BIFTOOL ([11, 9]). Table 1 lists the mesh characteristics  $m$ ,  $N_e$ ,  $\ell$  and  $N$ . Note that the Newton updates of  $T$  and  $\tau_k$  can force a change of  $\ell$  requiring a change

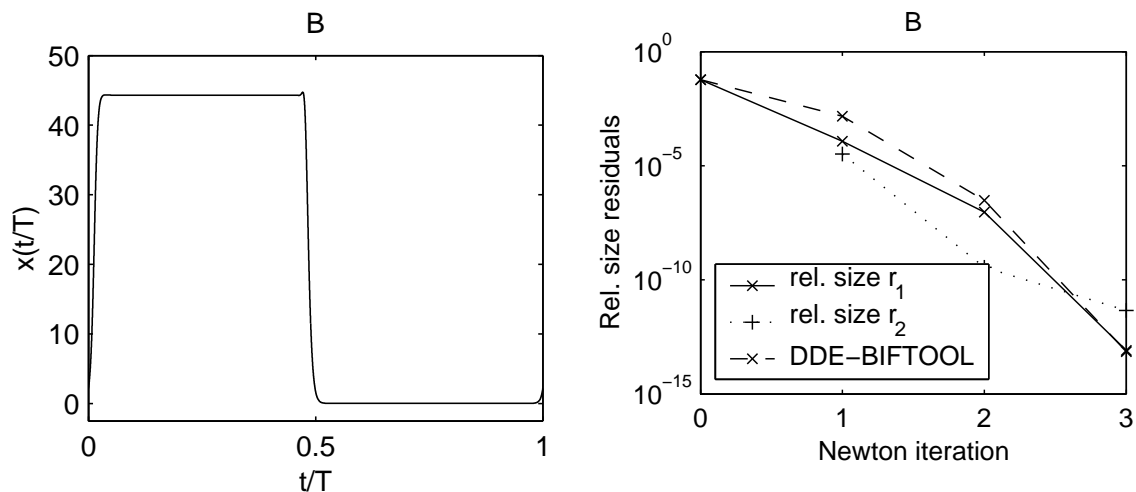


Figure 3: For periodic solution B: Left: The orbit  $x(t/T)$ . Right: The relative size of the residuals vs. the Newton iteration. For the Newton-Picard collocation method: the relative size of  $r_1$  ( $\times$ , solid line) and  $r_2$  ( $+$ , dotted line) in (17a) and (17b), respectively. For DDE-BIFTOOL: the relative size of its collocation residual ( $\times$ , dashed line).

	$m$	$N_e$	$\ell$	$N$
A.I	18	108	3	20
A.II	144	864	3	20
B	1024	3072	1120	3481

Table 1: The values of the mesh characteristics  $m$ ,  $N_e$ ,  $\ell$  and  $N$ .

of the length of  $u_{..0}$  and  $u_{..1}$ . However, this did not happen for these three periodic solutions. Remark the use of a large number of mesh intervals,  $m$ , in case of periodic solution B. This fine discretization is necessary to capture characteristics of the orbit such as the steep gradients and the little peak at  $t/T \approx 0.5$  in Fig. 3 (left).

Table 2 gives the dominant Floquet multipliers. For each example, we list both the eigenvalues obtained by the subspace iteration in the last Newton iteration — i.e., before the final Newton update — as well as refined eigenvalues obtained by performing a few extra subspace iterations after computing the periodic solution. The underlined digits are the ones that correspond to the Floquet multipliers computed by DDE-BIFTOOL. The latter builds the monodromy matrix explicitly and uses QR to compute all its eigenvalues. Clearly, for these examples, the approximations of the dominant Floquet multipliers obtained as a by-product of the Newton-Picard collocation method are accurate enough to assess the stability. However, for a more precise location of bifurcation points, it is better to use refined eigenvalues. At a limit point of cycles, e.g., the double eigenvalue at one is badly conditioned. Hence, refinement is absolutely necessary to obtain enough accuracy. Also note that the computed trivial multiplier has to be interpreted carefully. Its accuracy (i.e., the deviation from one) is not always comparable to the accuracy of the computed periodic solution nor to the accuracy of the other multipliers ([17]).

	$ \mu_1 $	$ \mu_2 $	$ \mu_3 $	$p$
A.I	1.000948	$4.577660 \times 10^{-1}$	$1.546415 \times 10^{-2}$ (*)	1
	<u>0.999997</u>	<u><math>4.595681 \times 10^{-1}</math></u>	<u><math>1.546822 \times 10^{-2}</math></u> (*)	
A.II	5.694530	1.000113		2
	<u>5.694558</u>	<u>1.000108</u>		
B	1.008251	$4.227757 \times 10^{-1}$	$1.447408 \times 10^{-1}$	1
	<u>0.999999</u>	<u><math>4.168833 \times 10^{-1}</math></u>	<u><math>1.649826 \times 10^{-1}</math></u>	

Table 2: Left: The moduli of the dominant Floquet multipliers where (\*) indicates an eigenvalue that belongs to a complex conjugated pair. For each periodic solution, the results from the Newton-Picard collocation method are listed in a row above the results from the computation afterwards with a sharper tolerance. The correct digits of these refined Floquet multipliers are underlined. Right:  $p$ , after the last subspace iteration.

We compared the convergence of the Newton iteration for the Newton-Picard collocation method and the method in DDE-BIFTOOL. The relative size of the residuals versus the Newton iteration for both methods is depicted in Fig. 4 and 3 (right). The residual  $r_2$  of the periodicity condition is zero before the first Newton iteration, since the initial orbit in Algorithm 5.2 is periodic. Remark that  $r_2$  vanishes after the last Newton iteration in case of periodic solution A.II (see Fig. 4, right) due to *numerical underflow*. For A.I and B on the other hand, the relative size of  $r_2$  in the last step is of the order of  $10^{-10}$ , the convergence threshold, while the relative size of  $r_1$  is much smaller. This is not unexpected. In the last Newton iteration, the Newton-Picard steps are stopped as soon as the convergence threshold for the Newton process is reached, rather than continued until the residual  $r_{\text{NP}}$  is small enough to guarantee quadratic convergence of the Newton iteration. However,  $\Delta u_{0 \triangleright 1}$  is computed exactly from (24). Therefore, we still obtain quadratic convergence to an orbit of the DDE system (measured by  $r_1$ ) but not to a periodic orbit (measured by  $r_2$ ). This also illustrates why we insisted on solving the shooting problem at each Newton iteration accurately enough to preserve quadratic convergence of  $r_1$  and  $r_2$ . Quadratic convergence to an orbit but slower convergence to a closed orbit implies that the method would behave more like a single shooting method.

Fig. 5 and Fig. 6 illustrate the convergence of the Newton-Picard steps in Algorithm 5.1. The numbering of the horizontal axis, the Newton-Picard steps, is restarted at every Newton iteration. All values are reported at the end of the step, where “step 0” actually denotes the values at the start of the first step. To gain more insight in the convergence behavior, we have computed  $r_c$  after each Newton-Picard update using (27), though this value is only required at the start of each Newton iteration. We also show  $r_{\text{NP}}$  (the residual of (29)) and its Q projection. Since we use zero starting values for Algorithm 5.1,  $r_{\text{NP}} = r_c$  at the start of each Newton iteration. Furthermore, we also show the estimate for the higher order terms used to stop the Newton-Picard steps when they have converged sufficiently to guarantee quadratic convergence of the Newton iterations. This estimate has to be compared with  $r_c$ , or even better with  $r_c - r_{\text{NP}}$ , at the end of each Newton iteration, but the latter value is not shown in the graphs. Because of its construction (which we did not discuss in the paper since it is very technical), the estimate is not available in the first Newton step. The figures show that the estimate is rather crude. However, as is confirmed by Fig. 3 (right) and Fig. 4, it is good enough to preserve the quadratic convergence of the Newton iterations. Note also that after the first Newton-Picard step,  $\|r_{\text{NP}}\|_2 \approx \|Qr_{\text{NP}}\|_2$  since  $Pr_{\text{NP}} \approx 0_{N \times 1}$ . Remark that the improvement in the last Newton-Picard step for example A.II is much larger than expected. Doing multiple Newton-Picard steps at each

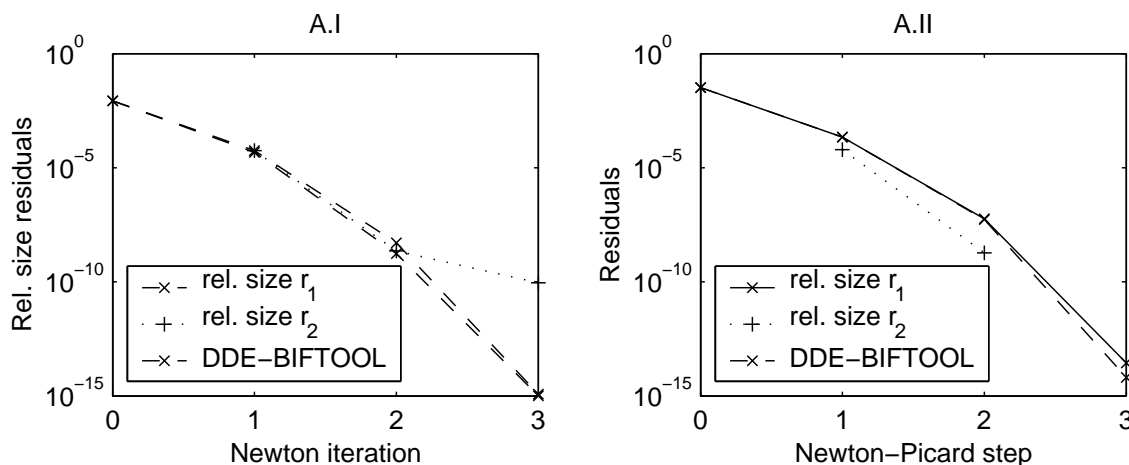


Figure 4: The relative size of the residuals vs. the Newton iteration for periodic solutions A.I (left) and A.II (right). For the Newton-Picard collocation method: the relative size of  $r_1$  ( $\times$ , solid line) and  $r_2$  ( $+$ , dotted line) in (17a) and (17b), respectively. For DDE-BIFTOOL: the relative size of its collocation residual ( $\times$ , dashed line).

Newton iteration was clearly useful in these cases. Otherwise, we would have needed a much better basis  $V_p$  to guarantee quadratic convergence.

The computational cost of our algorithm mainly depends on the number of matrix-vector products with the monodromy matrix  $M$ . This number depends on the spectrum of  $M$  and is almost independent of the mesh size ([18, 16]). For the computation of periodic solutions A.I, A.II and B, 48, 25, and 83 matrix-vector products with  $M$  were required, respectively. For periodic solution A.I, we needed 16 matrix-vector products during the iterative basis computation and 32 during the Picard iteration. In this case,  $|\mu_2| = 4.5957 \times 10^{-1}$ . Dropping  $\rho$  to 0.4 reduces the number of Picard iterations to 11 while keeping the number of matrix-vector products for the basis computation the same. However, we should not conclude from this that taking a smaller value of  $\rho$  is always better. Using a larger basis size will always improve the convergence of the Picard iterations, but the cost of the computation of the basis will often increase.

Finally, we also compared the computation time of our Matlab implementation of the Newton-Picard collocation method with the DDE-BIFTOOL implementation. Our algorithm suffers a lot more from Matlab-related overhead than DDE-BIFTOOL, since the latter can make more use of Matlab built-in commands. Nevertheless, timing results can give a rough idea about the applicability of a method. Table 3 lists the average CPU time over a few runs on a 1.7 GHz Pentium 4 computer. The second and third column of this table contain the time needed to compute a periodic solution, while the two rightmost columns give the total time to obtain an accurate periodic solution *and* accurate Floquet multipliers. For DDE-BIFTOOL (column 4), this is the sum of the time needed to compute the periodic solution and the time to compute the Floquet multipliers by constructing the monodromy matrix  $M$  explicitly and computing all its eigenvalues using QR. For the Newton-Picard collocation method (column 5), the second term in the sum is the time needed for the additional subspace iterations to improve the final basis. Those iterations were continued until  $\|QMV_p\|_2 \leq 10^{-5}$ . In all these cases, our Newton-Picard collocation algorithm outperforms DDE-BIFTOOL. The difference is most

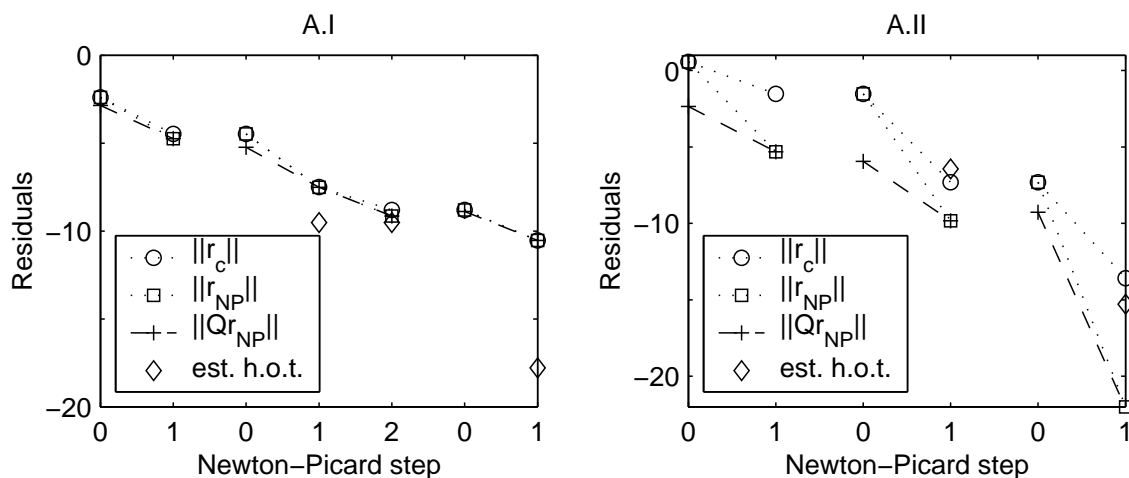


Figure 5: For periodic solutions A.I (left) and A.II (right):  $\|r_c\|_2$  ( $\circ$ , dotted line),  $\|r_{NP}\|_2$  ( $\square$ , dotted line),  $\|Qr_{NP}\|_2$  ( $+$ , dashed line) and an estimation of the higher-order terms ( $\diamond$ ) vs. the Newton-Picard step.

	D.-B. $P$	NP-c $P$	D.-B. $P+F$	NP-c $P+F$
A.I	6.8	1.7	$6.8 + 0.2 = 7.0$	$1.7 + 0.1 = 1.8$
A.II	23.8	13.1	$23.8 + 4.9 = 28.7$	$13.1 + 0.3 = 13.4$
B	610.5	126.4	$610.5 + 689.1 = 1299.6$	$126.4 + 8.9 = 135.3$

Table 3: CPU time in seconds for DDE-BIFTOOL (“D.-B.”) and the Newton-Picard collocation method (“NP-c”) for the computation of a periodic solution (“ $P$ ”) and the computation of both a periodic solution and the Floquet multipliers (“ $P+F$ ”).

significant for periodic solution B, where a fine discretization was needed. The computation time in DDE-BIFTOOL grows as  $N_e^3 = m^3 n^3 d^3$ . However, the cost for one matrix-vector product with  $M$  behaves like  $m \cdot n^3 \cdot d^3$ . The required LU factors have to be computed only once per Newton iteration. Hence, each additional matrix-vector product in the same Newton iteration only incurs an extra cost of order  $mn^2d^2$ . The number of matrix-vector products needed depends on the spectrum of  $M$  and not on  $m$ ,  $n$  or  $d$ . Our method has clear advantages for larger values of  $m$ . Note that the  $nd \times nd$  systems that are solved when computing matrix-vector products have a special structure if the partial derivatives  $\partial f / \partial x^k$  have a special structure, e.g., if the DDE system is the result of a space discretization of a delay PDE. This structure can sometimes be exploited to further reduce the cost in these cases.

This argument also holds for the computation of the (dominant) Floquet multipliers, when DDE-BIFTOOL currently uses an algorithm with  $\mathcal{O}(\max(N_e^3, N^3))$  complexity.

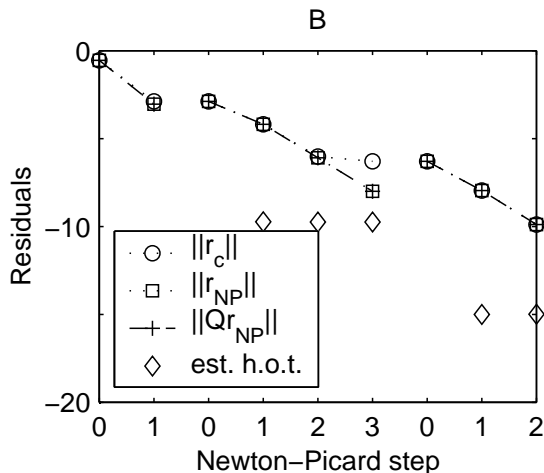


Figure 6: For periodic solution B:  $\|r_c\|_2$  ( $\circ$ , dotted line),  $\|r_{NP}\|_2$  ( $\square$ , dotted line),  $\|Qr_{NP}\|_2$  ( $+$ , dashed line) and an estimation of the higher-order terms ( $\diamond$ ) vs. the Newton-Picard step.

## 8 Conclusions.

In this paper we have developed a Gauss-Legendre Runge-Kutta collocation method with an iterative linear system solver to compute periodic solutions of a system of autonomous DDEs efficiently. Two ideas lie at the core of this development.

First, we have shown that the nonlinear collocation system can be solved using a Newton iteration at the continuous level combined with single shooting for the resulting linear BVPs. In this single shooting procedure, we used the implicit Runge-Kutta time integrator corresponding to the Runge-Kutta collocation scheme on the same mesh. This approach combines the advantages of collocation and single shooting methods without most of the disadvantages of either method. In particular, since single shooting is only done for a linear BVP, it does not suffer from the typical lack of robustness in some cases, e.g., the computation of unstable periodic solutions. Single shooting requires the solution of a bordered system with a Jacobian matrix whose main block is just a shift of the monodromy matrix. Hence, contrary to the linearized collocation requirements, the spectral properties of this main block are well understood and can easily be exploited by iterative solvers. Moreover, matrix-vector products with the monodromy matrix can be computed efficiently, requiring only the solution of  $m$  linear systems of size  $nd \times nd$  each. We have worked out the algorithm for a Gauss-Legendre Runge-Kutta collocation variant, but the procedure can be generalized to many other collocation and finite difference schemes.

Second, we have shown that the iterative Newton-Picard method is well suited to solve the linearized single shooting systems. The Newton-Picard method was originally conceived to approximately solve the linearized systems in a Newton process, resulting in overall linear convergence. We have reworked the method, in order to use multiple Newton-Picard steps per Newton iteration. In this manner, each linear system is solved with sufficient accuracy to maintain quadratic convergence of the Newton iteration. This is important since otherwise the resulting method would behave more like a single shooting method and the robustness of the collocation scheme might be lost. We have discussed the convergence of the method and exploited this analysis to tune the algorithm to reduce the number of matrix-vector products with the monodromy matrix. These improvements are not only useful for our

Newton-Picard collocation algorithm but can also be used in other implementations of the Newton-Picard method. Also note that the Newton-Picard method produces good estimates of the dominant Floquet multipliers as a by-product, making the method particularly attractive for bifurcation analysis.

Two models were presented as test cases to demonstrate the robustness and efficiency of the algorithm and the effectivity of the modifications to the Newton-Picard method.

Our Newton-Picard collocation method is most useful when a fine discretization is used or for large systems of DDEs. In both cases, using a direct linear system solver for the linearized collocation system becomes prohibitively expensive — even when restricting the unknowns to  $[0, 1]$ , as DDE-BIFTOOL does. Our method is also interesting if  $\tau/T$  is large, since then the monodromy matrix, needed in DDE-BIFTOOL to compute the Floquet multipliers, becomes large. However, the Newton-Picard method will only be efficient if the number of Floquet multipliers close to or outside the unit circle is small. In practice, this assumption is satisfied by many problems. Note that the method also needs to compute an approximate basis for the subspace corresponding to the dominant Floquet multipliers. However, when used in a continuation process, a good starting value for this basis is available from the computation of the previous periodic solution. Therefore, the Newton-Picard collocation method is particularly suited to compute branches of periodic solutions of large-scale problems with time delays. Our method can also be used for certain systems without time delays, such as parabolic PDEs or certain large ODE systems. We also expect that the method can be extended, like the Newton-Picard single shooting method ([10]), to compute bifurcation points and continue branches of bifurcation points.

## Acknowledgement.

The authors would like to thank Prof. Dirk Roose for his helpful comments concerning the presentation of the material.

## References

- [1] U. M. ASCHER, R. M. MATTHEIJ, AND R. D. RUSSELL, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, vol. 13 of Classics in Applied Mathematics, SIAM, 1995.
- [2] Z. BAI, J. DEMMEL, J. DONGARRA, A. RUHE, AND H. VAN DER VORST, eds., *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, vol. 11 of Software, Environments, and Tools, SIAM, 2000.
- [3] J. BÉLAIR AND M. MACKEY, *A model for the regulation of mammalian platelet production*, Ann. N.Y. Acad. Sci., 504 (1987), pp. 280–282.
- [4] O. DIEKMANN, S. VAN GILS, S. VERDUYN LUNEL, AND H.-O. WALTHER, *Delay Equations*, vol. 110 of Applied Mathematical Sciences, Springer-Verlag, 1995.
- [5] E. DOEDEL, A. CHAMPNEYS, T. FAIRGRIEVE, Y. KUZNETSOV, B. SANDSTEDTE, AND X.-J. WANG, *AUTO97: Continuation and bifurcation software for ordinary differential equations*, technical report, Dept. of Computer Science, Concordia University, 1998.

- [6] E. DOEDEL, H. KELLER, AND J. KERNÉVEZ, *Numerical Analysis and Control Of Bifurcation Problems, Part II : Bifurcation in Infinite Dimensions*, Int. J. Bifurcation Chaos, 1 (1991), pp. 745–772.
- [7] E. DOEDEL AND J. KERNÉVEZ, *AUTO: Software for continuation and bifurcation problems in ordinary differential equations.*, Applied Mathematics Report, California Institute of Technology, Pasadena, U.S.A., 1986.
- [8] K. ENGELBORGHES, *Numerical bifurcation analysis of delay differential equations*, PhD thesis, Department of Computer Science, K.U.Leuven, Leuven, Belgium, May 2000.
- [9] K. ENGELBORGHES AND E. DOEDEL, *Stability of piecewise polynomial collocation for computing periodic solutions of delay differential equations*, Numer. Math., 91 (2002), pp. 627–648.
- [10] K. ENGELBORGHES, K. LUST, AND D. ROOSE, *Direct computation of period doubling bifurcation points of large-scale systems of ODEs using a Newton-Picard method*, IMA J. Numer. Anal., 19 (1999), pp. 525–547.
- [11] K. ENGELBORGHES, T. LUZYANINA, K. IN ’T HOUT, AND D. ROOSE, *Collocation methods for the computation of periodic solutions of delay differential equations*, SIAM J. Sci. Comput., 22 (2000), pp. 1593–1609.
- [12] K. ENGELBORGHES, T. LUZYANINA, AND D. ROOSE, *Numerical bifurcation analysis of delay differential equations using DDE-BIFTOOL*, ACM Trans. Math. Softw., 28 (2002), pp. 1–21.
- [13] K. ENGELBORGHES, T. LUZYANINA, AND G. SAMAËY, *DDE-BIFTOOL v. 2.00 user manual: a Matlab package for numerical bifurcation analysis of delay differential equations*, Report TW 330, Department of Computer Science, K.U.Leuven, Leuven, Belgium, Oct. 2001.
- [14] J. HALE AND S. M. VERDUYN LUNEL, *Introduction to Functional Differential Equations*, vol. 99 of Applied Mathematical Sciences, Springer-Verlag, 1993.
- [15] K. LUST, *Numerical bifurcation analysis of periodic solutions of partial differential equations*, PhD thesis, Department of Computer Science, K.U.Leuven, Leuven, Belgium, Dec. 1997.
- [16] K. LUST, D. ROOSE, A. SPENCE, AND A. CHAMPNEYS, *An adaptive Newton-Picard algorithm with subspace iteration for computing periodic solutions*, SIAM J. Sci. Comput., 19 (1998), pp. 1188–1209.
- [17] T. LUZYANINA AND K. ENGELBORGHES, *Computing Floquet multipliers for functional differential equations*, Int. J. Bifurcation Chaos, 11 (2002), pp. 737–753.
- [18] T. LUZYANINA, K. ENGELBORGHES, K. LUST, AND D. ROOSE, *Computation, continuation and bifurcation analysis of periodic solutions of delay differential equations*, Int. J. Bifurcation Chaos, 7 (1997), pp. 2547–2560.
- [19] D. ROOSE, K. LUST, A. CHAMPNEYS, AND A. SPENCE, *A Newton-Picard shooting method for computing periodic solutions of large-scale dynamical systems*, Chaos Solitons Fractals, 5 (1995), pp. 1913–1925.
- [20] R. SEYDEL, *Practical Bifurcation and Stability Analysis: From Equilibrium to Chaos*, vol. 5 of Interdisciplinary Applied Mathematics, Springer-Verlag, second edition ed., 1994.

- [21] L. SHAYER AND S. CAMPBELL, *Stability, bifurcation and multistability in a system of two coupled neurons with multiple time delays*, SIAM J. Appl. Math., 61 (2000), pp. 673–700.
- [22] G. SHROFF AND H. KELLER, *Stabilization of unstable procedures: the recursive projection method*, SIAM J. Numer. Anal., 30 (1993), pp. 1099–1120.