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

Verheyden, Koen; Lust, Kurt

Published in:
Bit numerical mathematics

DOI:
10.1007/s10543-005-0013-4

Publication date:
2005

BIT Numerical Mathematics (2005) 45: 605–625
DOI: 10.1007/s10543-005-0013-4

A NEWTON-PICARD COLLOCATION METHOD FOR PERIODIC SOLUTIONS OF DELAY DIFFERENTIAL EQUATIONS*

KOEN VERHEYDEN[1] and KURT LUST[2,1,**]

[1] *Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Heverlee-Leuven, Belgium. email: Koen.Verheyden@cs.kuleuven.ac.be*

[2] *Institute for Mathematics and Computing Science, University of Groningen, PO Box 800, 9700 AV Groningen, The Netherlands. email: K.W.A.Lust@math.rug.nl*

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 exploit the equivalence of the linearized collocation system and the discretization of the linearized periodic boundary value problem (BVP). This linear BVP is solved using a variant of the Newton-Picard method [Int. J. Bifurcation Chaos, 7 (1997), pp. 2547–2560]. This method combines a direct method in the low-dimensional subspace of the weakly stable and unstable modes with an iterative solver in the high-dimensional 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.

AMS subject classification (2000): 65J15, 65P30, 65Q05.

Key words: delay differential equation, periodic solution, collocation, Newton-Picard, numerical bifurcation analysis.

1 Introduction.

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

$$x'(t) = f\left(x(t), x(t-\tau_1), \ldots, x(t-\tau_\kappa)\right), \tag{1.1}$$

where $x(t) \in \mathbb{R}^n$ and τ_k, $k = 1, \ldots, \kappa$, are constant, non-negative delays. Here f is a two times continuously differentiable function from $\mathbb{R}^{n(\kappa+1)}$ into $\mathbb{R}^n$.

* Received April 2003. Revised March 2005. Communicated by Gustaf Söderlind.

** Koen Verheyden is a Research Assistant of the Fund for Scientific Research—Flanders (Belgium) and during this research Kurt Lust was a Postdoctoral Fellow of this institution. This paper presents research results of the Belgian Programme on Interuniversity Attraction Poles, initiated by the Belgian Federal Science Policy Office. The scientific responsibility remains with its authors.

We search for periodic orbits $x^\star(t)$ of (1.1) with a nonzero *period* $T^\star$. That is, orbits that are non-constant (i.e., not an equilibrium point) and for which

$$x^\star(t + T^\star) = x^\star(t), \quad \text{for all } t \in \mathbb{R}. \tag{1.2}$$

Collocation is a robust technique to solve this periodic BVP. The linearized systems in the Newton iteration are usually sparse. However, contrary to the ODE case, it is often impossible to exploit their structure using a direct solver. Therefore, the implementation in the bifurcation analysis package DDE-BIFTOOL [11, 13, 12] suffers from a high computational and memory cost in the case of large systems of DDEs.

The novelty of this paper lies in three ideas. First, we rework the linearized collocation requirements to obtain a linear system that is suited for an iterative solver. This procedure is based on the correspondence between the linearization of our collocation scheme and the discretization of the linearized BVP. This is a well-known fact for ODEs [1, 21], but we show that it is also the case for DDE systems.

Second, we solve the linearized BVP using the Newton-Picard single-shooting method [18, 15, 16]. This avoids the robustness problems typically associated with shooting methods. The resulting procedure is equivalent to condensation of the linearized collocation system. The Newton-Picard method is an efficient method to solve nonlinear systems whose linearization can be written as

$$\begin{bmatrix} M - I & B \\ C^T & D \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta \lambda \end{bmatrix} = - \begin{bmatrix} r \\ s \end{bmatrix} \tag{1.3}$$

with $M \in \mathbb{R}^{N \times N}$, $B, C \in \mathbb{R}^{N \times k}$, N very large and k small (typically 1 or 2). The method assumes that the eigenvalues of M are strongly clustered about 0 and that only few eigenvalues are close to or larger than 1 in modulus. It combines a direct linear system solver in the subspace of the weakly stable and unstable modes of M with a fixed-point (or Picard) iteration in the high-dimensional orthogonal complement.

Third, we improve the Newton-Picard method to increase its efficiency and reliability. Rather than using the Newton-Picard method to solve a nonlinear system as in our previous work, we will now use it to solve the linearized collocation equations, i.e., the single Newton-Picard loop is replaced with a two-level structure with an outer Newton iteration and an inner Newton-Picard iteration. This allows to retain the quadratic convergence of the Newton iteration, contrary to earlier implementations, and to save on the costs for the basis computations.

The resulting algorithm is equivalent (in exact arithmetic) to a collocation method. Our approach combines the robustness of collocation with the efficiency of the Newton-Picard single shooting method [18].

The plan of the paper is as follows. In Section 2 we describe our particular variant of the Runge-Kutta collocation scheme. Section 3 discusses the linearization of the collocation scheme and the solution of the resulting linear BVP using single shooting, which reduces the linearized system to one of the form (1.3). Section 4 completes the presentation of the Newton-Picard collocation method

by integrating the Newton-Picard method in the procedure. In Section 5 we present numerical results. Finally, Section 6 summarizes the main conclusions from this work.

2 Discretization by a Runge-Kutta collocation scheme.

2.1 The periodic boundary value problem.

Let $\tau_0 := 0$ for notational convenience. As in [6, 11], we first rescale time by a factor $1/T$ such that the period is one in the transformed time. Instead of solving (1.1), we will solve the transformed equation

$$x'(t) = \mathcal{F}(x,T)(t) := T f\left(x(t-\tau_0/T), \ldots, x(t-\tau_\kappa/T)\right) \tag{2.1}$$

throughout the paper. For simplicity of notations, we don't introduce new symbols for the transformed variables.

Define $\tau := \max_k \tau_k$, the maximal delay. The periodic orbit is uniquely determined by the BVP

$$x'(t) = \mathcal{F}(x,T)(t), \quad \text{for } t \in [0,1], \tag{2.2a}$$

$$x(\theta+1) - x(\theta) = 0, \quad \text{for } \theta \in [-\tau/T, 0], \tag{2.2b}$$

$$\alpha(x) = 0 \tag{2.2c}$$

with unknowns x and T. It is important to note that the periodicity condition (2.2b) must enforce the equality of the *starting* and the *final function segment* of length τ/T, instead of just $x(1) = x(0)$ in the ODE case. This stems from the fact that the state space of (2.1) is $\mathcal{C}([-\tau/T, 0], \mathbb{R}^n)$. The *phase condition* (2.2c) removes translational invariance. We use the integral phase condition

$$\alpha(x) := \int_0^1 x^T(t) \frac{\mathrm{d}}{\mathrm{d}t} x^\circ(t)\, \mathrm{d}t = 0 \tag{2.3}$$

which minimizes the phase shift of $x(t)$ with respect to some reference solution $x^\circ(t)$, cf. [7]. This condition reduces the need for frequent remeshing.

2.2 Runge-Kutta collocation.

Let $\{0 = t_0 < t_1 < \ldots < t_m = 1\}$ be a mesh on $[0,1]$. This mesh is periodically extended to the left to obtain a mesh $\{t_{-\ell}, \ldots, t_m = 1\}$ on $[t_{-\ell}, 1] \supseteq [-\tau/T, 1]$. We determine ℓ such that $t_{-\ell} \leqslant -\tau/T < t_{-\ell+1}$. Let $\mathbb{P}_d^n|_{\mathsf{D}}$ denote the set of polynomials of degree d or less that map $\mathsf{D} \subseteq \mathbb{R}$ into $\mathbb{R}^n$. We approximate a function $x \in \mathcal{C}([t_{-\ell}, 1], \mathbb{R}^n)$ by an element u of the space of piecewise polynomials

$$\left\{ u \in \mathcal{C}\big([t_{-\ell}, 1], \mathbb{R}^n\big) \;:\; u|_{[t_i, t_{i+1}]} \in \mathbb{P}_d^n|_{[t_i, t_{i+1}]},\ i = -\ell, \ldots, m-1 \right\}. \tag{2.4}$$

We will represent the elements of this space in a convenient way. Let $\Delta t_i := t_{i+1} - t_i$ and $t_{i+\frac{j}{d}} := t_i + \frac{j}{d}\Delta t_i$ as well as $u_{i+\frac{j}{d}} := u(t_{i+\frac{j}{d}})$ for $i = -\ell, \ldots, m-1$

and $j = 0, \ldots, d$. By abuse of notation, we will denote the vector that groups the values $u_{i+\frac{j}{d}}$ by u. We also define the *starting* and *final* state vector and the *trajectory* vector by

$$u_s := \begin{bmatrix} u_{-\ell} \\ \vdots \\ u_{i+\frac{j}{d}} \\ \vdots \\ u_0 \end{bmatrix}, \quad u_f := \begin{bmatrix} u_{m-\ell} \\ \vdots \\ u_{i+\frac{j}{d}} \\ \vdots \\ u_m \end{bmatrix} \in \mathbb{R}^N, \quad u_t := \begin{bmatrix} u_{\frac{1}{d}} \\ \vdots \\ u_{i+\frac{j}{d}} \\ \vdots \\ u_m \end{bmatrix} \in \mathbb{R}^{N_t}, \tag{2.5}$$

where $N := n(\ell d+1)$ and $N_t := nmd$. Clearly, $u = (u_s,\, u_t)$ and u_f consists of the last N components of the vector u. Let ψ_j, $j = 0, \ldots, d$, denote the Lagrange polynomial of degree d which is one in j/d and zero in all other points from the set $\{0, 1/d, \ldots, 1\}$. For the restriction of the spline to a single mesh interval $[t_i, t_{i+1}]$, we use these polynomials rescaled to this mesh interval as the basis functions, i.e.,

$$u(t) = \sum_{j=0}^{d} u_{i+\frac{j}{d}} \psi_j \left(\frac{t - t_i}{\Delta t_i} \right), \quad t \in [t_i, t_{i+1}].$$

Our approximation to a solution $(x^\star, T^\star)$ to (2.2) is (u, T), which is obtained by imposing collocation requirements on (2.2a) and discretizing (2.2b) and (2.2c). As in AUTO [5, 6] and DDE-BIFTOOL, we use the *collocation points* $\bigcup_{i=0}^{m-1} \bigcup_{\nu=1}^{d} \{c_{i,\nu} := t_i + c_\nu \Delta t_i\}$ on $[0, 1]$, with c_ν the zeros of the *Legendre polynomial* of degree d. Note that for this *Gauss-Legendre collocation scheme* the accuracy of the orbit is of order h^{d+1} [8, 11, 9], which is different from the ODE case. The integral phase condition (2.3) is computed exactly by using a Gauss quadrature rule of degree d on each mesh interval $[t_i, t_{i+1}]$. Hence, the discretization of (2.2) becomes

$$r_1(u, T) = 0_{N_t \times 1}, \tag{2.6a}$$

$$r_2(u_s, u_f) := u_f - u_s = 0_{N \times 1}, \tag{2.6b}$$

$$\alpha(u_0, u_t) = 0, \tag{2.6c}$$

where (2.6a) groups the md vector-valued collocation requirements

$$\mathcal{F}(u, T)(c_{i,\nu}) - u'(c_{i,\nu}) = 0,$$

for $i = 0, \ldots, m-1$ and $\nu = 1, \ldots, d$. In order to evaluate $u(c_{i,\nu} - \tau_k/T)$, we first choose the index $\hat{\imath}_{i,\nu,k}$ from the set $\{-\ell, \ldots, m-1\}$ such that $t_{\hat{\imath}_{i,\nu,k}} \leqslant c_{i,\nu} - \tau_k/T < t_{\hat{\imath}_{i,\nu,k}+1}$. Remark that $\hat{\imath}_{i,\nu,0} \equiv i$, since $\tau_0 = 0$. Next, we evaluate

$$u(c_{i,\nu} - \tau_k/T) = \sum_{j=0}^{d} \psi_j \left(\frac{(c_{i,\nu} - \tau_k/T) - t_{\hat{\imath}_{i,\nu,k}}}{\Delta t_{\hat{\imath}_{i,\nu,k}}} \right) u_{\hat{\imath}_{i,\nu,k}+\frac{j}{d}}. \tag{2.7}$$

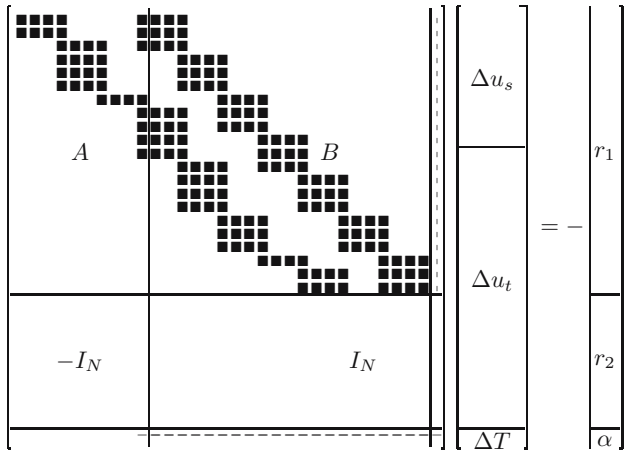

Figure 2.1: The structure of the linearized collocation system for one delay ($\kappa = 1$) that is smaller than the (approximate) period T, using a non-equidistant mesh with $m = 7$ mesh intervals and piecewise polynomials of degree $d = 3$. Here, the extended mesh contains $\ell = 3$ additional mesh intervals. (A black box represents a $n \times n$ block.)

2.3 Linearization.

The nonlinear collocation system (2.6) has $N + N_t + 1$ 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 Figure 2.1. Let us first specify its entries. The first N_t rows are the linearization of (2.6a), i.e.,

$$\sum_{j=0}^{d} \left(T \left(\sum_{k=0}^{\kappa} \psi_j \left(\frac{(c_{i,\nu} - \tau_k/T) - t_{\hat{i}_{i,\nu,k}}}{\Delta t_{\hat{i}_{i,\nu,k}}} \right) \frac{\partial f}{\partial x^k}(\cdot) \Delta u_{\hat{i}_{i,\nu,k}+\frac{j}{d}} \right) - \frac{\psi_j'(c_\nu)}{\Delta t_i} \Delta u_{i+\frac{j}{d}} \right)$$

$$+ \left(f(\cdot) + \frac{1}{T} \sum_{k=1}^{\kappa} \frac{\tau_k}{\Delta t_{\hat{i}_{i,\nu,k}}} \frac{\partial f}{\partial x^k}(\cdot) \sum_{j=0}^{d} \psi_j' \left(\frac{(c_{i,\nu} - \tau_k/T) - t_{\hat{i}_{i,\nu,k}}}{\Delta t_{\hat{i}_{i,\nu,k}}} \right) u_{\hat{i}_{i,\nu,k}+\frac{j}{d}} \right) \Delta T$$

$$= - \left(T f(\cdot) - u'(c_{i,\nu}) \right), \quad \text{for } i = 0, \ldots, m-1, \quad \nu = 1, \ldots, d, \tag{2.8}$$

where $f(\cdot)$ and its partial derivatives are evaluated at $(x^0, \ldots, x^\kappa) = \big(u(c_{i,\nu}), \ldots, u(c_{i,\nu} - \tau_\kappa/T)\big)$. Let A, B and $r_{1,T}$ denote the partial derivatives of r_1 w.r.t. u_s, u_t and T respectively. Then (2.8) can be written as

$$A\Delta u_s + B\Delta u_t + r_{1,T}\Delta T = -r_1. \tag{2.9a}$$

The second block row, consisting of N rows,

$$\Delta u_f - \Delta u_s = -r_2, \tag{2.9b}$$

comes from the periodicity condition (2.6b). If the maximal delay τ 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 (2.6c),

$$\alpha_0 \Delta u_0 + \alpha_t \Delta u_t + \alpha_T \Delta T = -\alpha, \tag{2.9c}$$

where α_0, α_t and α_T are partial derivatives w.r.t. u_0, u_t, respectively T. Remark that $\alpha_T \equiv 0$ for the integral phase condition that we use.

Consider the main block $[A\ B]$ of size $N_t \times (N + N_t)$ of the Jacobian matrix. If there were no delays (i.e., $\kappa = 0$) then $N = n$ and $[A\ B]$ would only consist of the rightmost, regularly shaped, staircase band of m subblocks of size $nd \times n(d+1)$ each. Each delay τ_k gives rise to an extra band. The shape of those bands is irregular unless an equidistant mesh is used.

Let us briefly compare our collocation variant to DDE-BIFTOOL's scheme [11, 13, 12]. In DDE-BIFTOOL, the 1-periodicity of the solution is explicitly used in the derivation of the collocation requirements. That is, every occurrence of $u(t)$ is replaced by $u(t \bmod 1)$ or, equivalently, $\hat{\imath}_{i,\nu,k}$ in (2.8) is replaced by $\hat{\imath}_{i,\nu,k} \bmod m$. The resulting linearized system can be obtained by eliminating $u_{-\ell}, u_{-\ell+1/d}, \ldots, u_{-1/d}$ from (2.9a) using the corresponding rows of (2.9b). Thus the bands in the main block, corresponding to the delays, are "folded" and become circular bands [12, Figure 2]. Clearly, it is difficult to exploit this structure in a direct solver. However, the same holds for (2.9). In the next sections, we work towards an efficient iterative solver for (2.9).

3 Solving a condensed linearized system.

In this section, we will manipulate the linearized collocation equations and condense the system to the form (1.3), suited for the Newton-Picard method. First, in Section 3.1, we consider the linearized system from a different viewpoint. Next, the monodromy matrix is defined in Section 3.2. In Section 3.3 we discuss the condensation of the linearized collocation system. We will show that this corresponds to a single shooting method for the linearization of the periodic BVP (2.2).

3.1 Discretization and linearization commute.

It is well-known that the order of discretization and linearization can be interchanged for the Gauss-Legendre scheme for an ODE BVP (see, e.g., [1, 21]). This is also the case for a DDE. The linearization of the periodic BVP (2.2) about a given trajectory $(x = u, T)$, is the linear BVP in the unknowns $(\Delta x, \Delta T)$

given by

$$
\begin{aligned}
&-(\Delta x)'(t) + \mathcal{F}_x(u,T)\Delta x(t) + \mathcal{F}_T(u,T)(t)\Delta T \\
&\qquad = -(\mathcal{F}(u,T)(t) - u'(t)), \quad \text{for } t \in [0,1],
\end{aligned} \tag{3.1a}
$$

$$
\Delta x(\theta+1) - \Delta x(\theta) = -(u(\theta+1) - u(\theta)), \quad \text{for } \theta \in [-\tau/T, 0], \tag{3.1b}
$$

$$
\alpha_x \Delta x + \alpha_T \Delta T = -\alpha(u). \tag{3.1c}
$$

Here, $\mathcal{F}_x$ and $\mathcal{F}_T$ are the Fréchet derivatives of $\mathcal{F}$.

Discretizing this linearized BVP using the Runge-Kutta collocation method from Section 2.2 also results in the linearized collocation system (2.9). That is, the solution procedure developed in Section 2 is equivalent to computing an approximate solution to the BVP (2.2) in the function space (2.4) by a Newton process for the BVP (2.2) where an approximate solution to the linearized system in each Newton iteration is computed by using our collocation scheme. This strategy is called *quasi-linearization* [1].

Remark that the "commutativity of discretization and linearization" holds for all discretization schemes that approximate x by an element u from a given vector space by using requirements that are linear in $u(\cdot)$ and $\mathcal{F}(u,T)(\cdot)$.

3.2 The monodromy matrix.

By definition, the action of the *monodromy operator* is time integration of the *variational equations* (i.e., (3.1a) with $\Delta T = 0$ and zero right hand side) about an (approximate) trajectory over $\Delta t = 1$ [14, 4]. It follows from the previous section that a discretization of these variational equations can be obtained from (2.9a) by setting $\Delta T = 0$ and $r_1 = 0_{N_t \times 1}$. That is, computing

$$
\Delta u_t = M_t \Delta u_s \quad \text{where } M_t := -B^{-1}A \in \mathbb{R}^{N_t \times N} \tag{3.2}
$$

for a given Δu_s is a numerical time integrator for the variational equations. Let us define the matrix $\tilde{I}_{K,L} := [0_{K\times(L-K)}\ I_K]$, i.e., $\tilde{I}_{K,L}v$ selects the last K components of $v \in \mathbb{R}^L$. From comparing (3.2) and (2.5), it is clear that the *monodromy matrix* M can be defined as follows:

$$
M := \tilde{I}_{N,N_t} M_t \quad \text{if } N \leqslant N_t, \tag{3.3a}
$$

and

$$
M := \begin{bmatrix} \tilde{I}_{N-N_t,N} \\ M_t \end{bmatrix} \quad \text{if } N > N_t. \tag{3.3b}
$$

The dominant eigenvalues of M are usually good approximations to the dominant Floquet multipliers which determine the stability of the periodic solution and the occurrence of bifurcation points.

By (3.3), the matrix-vector product with the monodromy matrix Mv can be easily obtained after computing $v_t := M_t v$. The latter can be computed by solving $Bv_t = -Av$ with a block version of forward substitution. Hence, the matrix-vector product Mv can be computed easily without constructing the (often large) dense matrix M. This favors the the use of iterative eigenvalue solvers like the Arnoldi method and orthogonal subspace iteration [2] to obtain the dominant Floquet multipliers. More importantly, it is the key element of the iterative solver for (2.9) presented in Section 4.

3.3 Condensation of the linearized system.

Let us briefly turn to AUTO and the ODE case. The linearized collocation system in AUTO is a special case of (2.9) with $N = n$ and with only the rightmost band of m subblocks [6, Figure 1.3]. This structure is exploited in AUTO by first eliminating all unknowns except Δu_0, Δu_m and ΔT in a two-phase elimination process which is similar to forward integration. The resulting system in Δu_0, Δu_m and ΔT is very close to the linearized system in a single shooting method. A system of the form (1.3) (with M the monodromy matrix) can be obtained by eliminating Δu_m using the periodicity constraint. Though this linear system solver is potentially unstable (especially for unstable periodic orbits), this does not seem to harm the convergence of the Newton iteration much.

We will now derive a similar procedure to reduce (2.9) to a similar system. Reordering (2.9a), premultiplying with B^{-1} and using (3.2), one obtains

$$\Delta u_t = M_t \Delta u_s - B^{-1} r_{1,T} \Delta T - B^{-1} r_1. \tag{3.4}$$

Computing Δu_t from (3.4), given Δu_s and ΔT, is equivalent to numerical time integration of (3.1a) (using a sequence of implicit Runge-Kutta steps with step sizes Δt_i) with initial condition Δu_s. To compute Δu_f (the final state of the integration) from (3.4), we need to distinguish between two cases. If $N \leqslant N_t$, we premultiply (3.4) with $\tilde{I}_{N,N_t}$ to obtain

$$\Delta u_f = M \Delta u_s - \tilde{I}_{N,N_t} B^{-1} r_{1,T} \Delta T - \tilde{I}_{N,N_t} B^{-1} r_1.$$

Substituting this expression in (2.9b), we obtain the equation

$$(M - I)\Delta u_s + z_T \Delta T = -r_c \tag{3.5}$$

with

$$r_c := r_2 - \tilde{I}_{N,N_t} B^{-1} r_1, \;\; z_T := -\tilde{I}_{N,N_t} B^{-1} r_{1,T}. \tag{3.6}$$

If $N > N_t$, the procedure is more involved. Now

$$\Delta u_f = \begin{bmatrix} \tilde{I}_{N-N_t,N} \\ M_t \end{bmatrix} \Delta u_s - \begin{bmatrix} 0_{(N-N_t)\times 1} \\ B^{-1} r_{1,T} \end{bmatrix} \Delta T - \begin{bmatrix} 0_{(N-N_t)\times 1} \\ B^{-1} r_1 \end{bmatrix}.$$

After substitution in (2.9b), we again obtain (3.5), but now

$$r_c := r_2 - \begin{bmatrix} 0_{(N-N_t)\times 1} \\ B^{-1}r_1 \end{bmatrix}, \; z_T := \begin{bmatrix} 0_{(N-N_t)\times 1} \\ -B^{-1}r_{1,T} \end{bmatrix}. \tag{3.7}$$

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

Substituting (3.4) in (2.9c), we obtain the condensed phase condition

$$a_u^T \Delta u_s + a_T \Delta T = -\alpha_c, \tag{3.8}$$

with

$$\begin{aligned} a_u^T &:= [0_{1\times(N-n)} \; \alpha_0] + \alpha_t M_t, \\ a_T &:= \alpha_T - \alpha_t B^{-1} r_{1,T}, \\ \alpha_c &:= \alpha - \alpha_t B^{-1} r_1. \end{aligned} \tag{3.9}$$

Note that the products with B^{-1} in (3.6), (3.7) and (3.9) can be computed by a block version of forward substitution, similarly to the computation of Mv in Section 3.2.

Hence, solving the linearized collocation system (2.9) is equivalent (in exact arithmetic) to solving the condensed system (3.5, 3.8) and then computing Δu_t from (3.4). System (3.5, 3.8) is now in a form suited for the Newton-Picard method. It is important to note that though the structure is identical, (3.5, 3.8) is not the linearized single shooting system for the nonlinear BVP (2.2).

4 The Newton-Picard method.

For DDEs, the monodromy matrix M has typically only a few eigenvalues close to or outside the unit circle. Moreover, as shown above, matrix-vector products with M can be computed without first constructing M. Therefore, the Newton-Picard method is a good option to solve (2.9). There are however a few new elements compared to the implementations discussed in [15, 16, 18]. First, the linearization of (2.6) does not result immediately in a system of the form (1.3), but the condensation process explained above is needed. This requires changes to the implementation. Second, as indicated in Section 1, we will also make further improvements to the Newton-Picard method. In previous work, the linearized system at each Newton iteration was solved only approximately using a combination of a direct and an iterative method. One such iteration step for the nonlinear system is called a Newton-Picard step. In this process, we lost quadratic convergence for the nonlinear iteration. In this paper, we use a two-level iteration: at each Newton step, we solve the linearized system with enough precision to retain the quadratic convergence by applying several Newton-Picard steps to the condensed system (3.5, 3.8). By doing so, we hope to lower the cost of the basis computation needed in the method. We will now work out this process in more detail.

4.1 The basic Newton-Picard step.

System (3.5, 3.8) can be solved with the iteration

1. $\Delta u_s^{(0)} = 0$, $\Delta T^{(0)} = 0$.
2. Loop for $\nu = 1, 2, \ldots$
   2.1. Compute an approximate solution to

$$\begin{bmatrix} M - I & z_T \\ a_u & a_T \end{bmatrix} \begin{bmatrix} \delta u_s^{(\nu)} \\ \delta T^{(\nu)} \end{bmatrix} = - \begin{bmatrix} r_{\mathrm{NP}}^{(\nu)} \\ \alpha_{\mathrm{NP}}^{(\nu)} \end{bmatrix} \tag{4.1}$$

   with

$$r_{\mathrm{NP}}^{(\nu)} := r_c + (M - I)\Delta u_s^{(\nu-1)} + z_T \Delta T^{(\nu-1)},$$
$$\alpha_{\mathrm{NP}}^{(\nu)} := \alpha_c + a_u \Delta u_s^{(\nu-1)} + a_T \Delta T^{(\nu-1)}.$$

   2.2. Update $\Delta u_s^{(\nu)} = \Delta u_s^{(\nu-1)} + \delta u_s^{(\nu)}$, $\Delta T^{(\nu)} = \Delta T^{(\nu-1)} + \delta T^{(\nu)}$.

The system (4.1) will be solved with enough accuracy such that the norm of the residual of (3.5, 3.8) decreases with a user-determined factor at every iteration (100 in our experiments).

To compute an approximate solution of (4.1), we proceed as follows. Let ρ be a user-determined threshold with $0 < \rho < 1$. Its optimal value depends on the spectrum of M, but values between 0.5 and 0.8 have usually proven to be adequate in our tests. Let $V_p \in \mathbb{R}^{N \times p}$ be an orthogonal matrix whose columns span the generalized eigenspace corresponding to all Floquet multipliers $\mu_1, \ldots, \mu_p$ with modulus larger than or equal to ρ. This is an invariant subspace of M. The threshold ρ should be such that p is a small number, typically $p < 10$. The matrix V_p is computed using orthogonal subspace iteration [19]. The matrix-vector products with M that are required for this method are again computed as in Section 3.2. Let $V_q \in \mathbb{R}^{N \times (N-p)}$ be an orthogonal matrix whose columns are a basis for the (high-dimensional) orthogonal complement of the column space of V_p. This space is not an invariant subspace of M since M is in general a nonnormal matrix. Furthermore, let

$$P := V_p V_p^T, \quad Q := V_q V_q^T = I_N - V_p V_p^T$$

be the orthogonal projectors onto the column space of V_p and its orthogonal complement respectively. Note that we need to avoid the explicit computation of V_q and the projectors P and Q. For the sake of simplicity we drop the superscripts of $\delta u_s^{(\nu)}$ and $\delta T^{(\nu)}$. Consider the decomposition

$$\delta u_s = V_p \delta \bar{p} + \delta q \quad \text{with} \quad \delta q := Q \delta u_s \tag{4.2}$$

of δu_s. After substituting (4.2) in (4.1) and premultiplying the first N equations with $[V_q \ V_p]^T$, one step of block-Gauss elimination reduces the system to the

$(p+1)\times(p+1)$-system

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T\,(z_T + M\delta q_T) \\ a_u^T V_p & a_T + a_u^T \delta q_T \end{bmatrix} \begin{bmatrix} \delta\bar{p} \\ \delta T \end{bmatrix} = - \begin{bmatrix} V_p^T\big(r_{\mathrm{NP}}^{(\nu)} + M\delta q_r\big) \\ \alpha_{\mathrm{NP}}^{(\nu)} + a_u^T \delta q_r \end{bmatrix}, \tag{4.3}$$

and

$$\delta q = \delta q_r + \delta q_T \delta T. \tag{4.4}$$

Here $\delta q_r = V_q \delta\bar{q}_r$, $\delta q_T = V_q \delta\bar{q}_T$ and $\delta\bar{q}_r$, $\delta\bar{q}_T$ are the solution of

$$\big(V_q^T (M - I) V_q\big) \begin{bmatrix} \delta\bar{q}_r & \delta\bar{q}_T \end{bmatrix} = - \begin{bmatrix} V_q^T r_{\mathrm{NP}}^{(\nu)} & V_q^T z_T \end{bmatrix}. \tag{4.5}$$

However, rather than solving (4.5) exactly, we compute approximations to δq_r and δq_T using the fixed point (or Picard) iteration

$$\delta q_r \leftarrow Q\big(M\delta q_r + r_{\mathrm{NP}}^{(\nu)}\big), \quad \delta q_T \leftarrow Q\big(M\delta q_T + z_T\big). \tag{4.6}$$

Finally, we set

$$\delta u_s = V_p \delta\bar{p} + \delta q_r + \delta T \delta q_T.$$

The convergence of this scheme depends in a complicated way on the accuracy of the basis V_p and the accuracy to which we solve (4.5). A discussion of this—in a slightly different context—can be found in [15].

After computing an approximate solution to (3.5, 3.8), Δu_t is computed exactly (up to small rounding errors) from (3.4) using a block version of forward substitution. As a result of this, at the end of the Newton step, r_1 will be negligible (depending on the rounding errors during the forward substitution), but r_2 will depend on the accuracy to which we solve (3.5, 3.8). Hence, (u_s, u_t) will converge quadratically to a trajectory of the DDE system, but it will only converge quadratically to a *periodic* orbit if (3.5, 3.8) is solved accurately enough. This will determine the stopping criterion for the Newton-Picard iterations.

We will now first present the overall algorithm and then elaborate on the various stopping criteria.

4.2 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 (2.6). In each Newton iteration, the large linearized system (2.9) (see Figure 2.1) is solved approximately by solving the condensed system (3.5, 3.8) approximately for Δu_s and ΔT and then computing Δu_t using (3.4). In every Newton iteration, the basis V_p is computed (or improved) using subspace iteration.

At the second level, we have the Newton-Picard loop outlined in the previous section. In every Newton-Picard step, the vectors δq_r and δq_T are computed using the Picard iteration (4.6), the third and innermost loop level.

The algorithm in Figure 4.1 presents these loops without their stopping criteria which are detailed in Section 4.3.

Algorithm: Newton-Picard collocation

1. Initialize the orbit, i.e., periodically extend $u(t)$ to a function on $[t_{-\ell}, 1]$.
2. Initialize the basis V_p.
3. Do until convergence *(outer level: Newton iterations)*
   - 3.1. Construct the condensed linearized system, i.e., compute z_T, a_u, a_T, r_c and α_c and prepare for efficiently computing matrix-vector products Mv.
   - 3.2. Set $\Delta u_s^{(0)} = 0_{N\times 1}$, $\Delta T^{(0)} = 0$, $r_{\text{NP}}^{(1)} = r_c$ and $\alpha_{\text{NP}}^{(1)} = \alpha_c$.
   - 3.3. Compute or improve the basis V_p *(using subspace iteration)*.
   - 3.4. Set $\delta q_r = 0_{N\times 1}$ and $\delta q_T = 0_{N\times 1}$.
   - 3.5. Do until convergence, for $\nu = 1, 2, \ldots$
     (second level: Newton-Picard steps)
     (inner level: Picard iterations)
     - 3.5.1. Picard iterations: $\delta q_r \leftarrow Q\big(M\delta q_r + r_{\text{NP}}^{(\nu)}\big)$.
     - 3.5.2. Picard iterations: $\delta q_T \leftarrow Q(M\delta q_T + z_T)$.
     - 3.5.3. Solve (4.3) directly.
     - 3.5.4. If *this Newton-Picard step is acceptable*, then
       Update $\Delta u_s^{(\nu)} = \Delta u_s^{(\nu-1)} + V_p\delta\bar{p} + \delta q_r + \delta q_T\delta T$,
       $\Delta T^{(\nu)} = \Delta T^{(\nu-1)} + \delta T$.
       Compute $r_{\text{NP}}^{(\nu+1)} = r_c + (M - I)\Delta u_s^{(\nu)} + \Delta T^{(\nu)} z_T$,
       $\alpha_{\text{NP}}^{(\nu+1)} = \alpha_c + a_u^T\Delta u_s^{(\nu)} + \Delta T^{(\nu)} a_T$.
       Set $\delta q_r = 0_{N\times 1}$ and $\delta q_T = 0_{N\times 1}$.
     - 3.5.5. Else
       Restart this Newton-Picard step: Go to 3.5.1.
     - 3.5.6. End if.
   - 3.6. End do. *(second level: Newton-Picard steps)*
   - 3.7. Update Δu_t using (3.4) with $\Delta u_s = \Delta u_s^{(\nu)}$ and $\Delta T = \Delta T^{(\nu)}$.
4. End do. *(outer level: Newton iterations)*

Figure 4.1: The Newton-Picard collocation algorithm.

4.3 The convergence criteria.

The stopping criteria for the basis and Picard iterations at the innermost loop level must guarantee convergence of the Newton-Picard procedure, while the stopping criterion for the Newton-Picard steps will guarantee quadratic convergence of the Newton iteration. The latter is stopped when the solution is deemed accurate enough.

4.3.1 The Newton iterations.

To test for convergence of the Newton iteration, any criterion used in a traditional Newton process is valid. Our stopping criterion checks if the relative residuals

$$\eta_1 := \frac{\|r_1\|_2}{T\sqrt{\sum_{i,\nu}\|f(\ldots,u(c_{i,\nu}-\tau_k/T),\ldots)\|_2^2}} \quad \text{and} \quad \eta_2 := \frac{\|r_2\|_2}{\|u_s\|_2} \tag{4.7}$$

both decrease below a user-given tolerance *RTOL*.

4.3.2 The Newton-Picard iterations.

The Newton-Picard steps are used to solve (3.5, 3.8) accurately enough to maintain quadratic convergence of the Newton iteration. To guarantee quadratic convergence, it is sufficient that the residual of (3.5, 3.8) at the end of the Newton-Picard steps is smaller than the higher-order terms neglected in the Newton linearization. Therefore we base our convergence criterion on an estimate of these terms. Remark that we do not consider the residual α_c of the condensed phase condition (3.8) because it will be zero up to rounding errors. Indeed, this condition is transformed to the last row of (4.3) which is solved by a direct method. Consider the previous ("old") Newton iteration. Let $r_{\mathrm{NP}}^{\mathrm{old}}$ denote the value of r_{NP} at the end of the Newton-Picard steps in the previous Newton iteration and $(\Delta u_s^{\mathrm{old}}, \Delta T^{\mathrm{old}})$ the update in that Newton iteration. One can show that at the end of the current Newton iteration, $r_c - r_{\mathrm{NP}}^{\mathrm{old}}$ is of second order in $(\Delta u_s^{\mathrm{old}}, \Delta T^{\mathrm{old}})$. Hence,

$$\frac{\left\|r_c - r_{\mathrm{NP}}^{\mathrm{old}}\right\|_2}{\left\|\Delta u_s^{\mathrm{old}}\right\|_2^2 + |\Delta T^{\mathrm{old}}|^2}$$

is a cheap estimate for the norm of the Hessian matrix of r_c about $(u_s^{\mathrm{old}}, T^{\mathrm{old}})$. From this information, at the end of each Newton-Picard step, the size of the higher-order contributions can be estimated by

$$\widehat{\text{h.o.t.}} := \frac{\left\|r_c - r_{\mathrm{NP}}^{\mathrm{old}}\right\|_2}{\left\|\Delta u_s^{\mathrm{old}}\right\|_2^2 + |\Delta T^{\mathrm{old}}|^2} \times \left(\|\Delta u_s\|_2^2 + |\Delta T|^2\right). \tag{4.8}$$

We also stop the Newton-Picard steps if we expect that the stopping criterion of the outer Newton iterations is met. For this, we estimate the Newton residual as $\|r_{\mathrm{NP}}\|_2 + \widehat{\text{h.o.t.}}$. At the end of each Newton-Picard step, we check if this estimate has decreased below the desired value of the Newton residual, i.e., its current value multiplied by $\mathrm{RTOL}/\max(\eta_1, \eta_2)$.

Summarizing, at the end of each Newton-Picard step, we check whether

$$\left\|r_{\mathrm{NP}}^{(\nu)}\right\|_2 \leqslant \widehat{\text{h.o.t.}} \quad \text{or} \quad \left\|r_{\mathrm{NP}}^{(\nu)}\right\|_2 + \widehat{\text{h.o.t.}} \leqslant \|r_c\|_2 \times \mathrm{RTOL}/\max(\eta_1, \eta_2). \tag{4.9}$$

If either condition is satisfied, we declare convergence of the Newton-Picard steps, execute the Newton correction, check for convergence of the Newton process and proceed with the next Newton iteration if necessary. Remark that the

estimate (4.8) is not available in the first Newton iteration. Then, the Newton-Picard steps are halted if $\|r_{\mathrm{NP}}^{(\nu)}\|_2 \leqslant 10^{-2} \times \|r_c\|_2$.

4.3.3 The innermost loop level.

Let us turn to the subspace iterations and Picard iterations within a certain Newton-Picard step.

The stopping criterion of the subspace iteration (and strategies to determine p) are detailed in [15, 16]. This criterion ensures among others that

$$\|QMV_p\|_2 = \left\|MV_p - V_p\left(V_p^T MV_p\right)\right\|_2, \tag{4.10}$$

—which is a measure for the accuracy of the basis V_p—decreases below a certain threshold. In our current implementation, this threshold is set to 10^{-2} or less.

The number of Picard iterations is adapted dynamically, contrary to the implementation of the Newton-Picard method in [16]. Our algorithm first executes a fixed number of Picard iterations and checks if the current Newton-Picard step can be accepted, see below. If not, additional Picard iterations are performed. For the sake of clarity, we did not fully write out this criterion in the algorithm in Figure 4.1. We now derive the mechanism behind it.

Consider the P and Q projections of $r_{\mathrm{NP}}^{(\nu+1)}$. By using (4.2) one obtains

$$Pr_{\mathrm{NP}}^{(\nu+1)} = Pr_{\mathrm{NP}}^{(\nu)} + P(M-I)(\delta p + \delta q_r + \delta q_T \delta T) + Pz_T \delta T \tag{4.11}$$

and

$$\begin{aligned} Qr_{\mathrm{NP}}^{(\nu+1)} &= Qr_{\mathrm{NP}}^{(\nu)} + Q(M-I)(\delta p + \delta q_r + \delta q_T \delta T) + Qz_T \delta T \\ &= QMV_p \delta\bar{p} + \left(Qr_{\mathrm{NP}}^{(\nu)} + Q(M-I)\delta q_r\right) \\ &\quad + \left(Qz_T + Q(M-I)\delta q_T\right)\delta T, \end{aligned} \tag{4.12}$$

where the values δp, δq_r, δq_T and δT are obtained from the ν^{th} Newton-Picard step. After premultiplication with V_p^T, the right-hand side of (4.11) comprises the first p equations of the small system (4.3). Since this system is solved with Gaussian elimination, we can safely assume that $\|Pr_{\mathrm{NP}}^{(\nu+1)}\|_2 = \|V_p^T r_{\mathrm{NP}}^{(\nu+1)}\|_2$ is zero up to the effect of rounding errors. Therefore, we only must be concerned with $Qr_{\mathrm{NP}}^{(\nu+1)}$, cf. (4.12).

The first term in the right hand side of (4.12) contains QMV_p, whose size can be decreased by performing more subspace iterations, cf. (4.10). The other terms in (4.12) are the residuals of the Picard iterations (4.6). It is desirable that the sizes of these three terms are balanced. Indeed, making one of those terms much smaller than the other one costs matrix-vector products with M while it does not improve the results. At the end of one Newton-Picard step, we compare the sizes of the Picard residuals to $\|QMV_p\delta\bar{p}\|_2$. If both Picard residuals are smaller, then the current Newton-Picard step is accepted. Else, some additional Picard iterations are performed in order to bring all three terms in (4.12) to the

same order of magnitude. In the algorithm in Figure 4.1, this is implemented by skipping the updates of Δu_s, ΔT, r_{NP} and α_{NP} and returning to the begin of the current Newton-Picard step. Remark that the number of additional Picard iterations needed can be predicted using an estimation of the convergence factor.

Note also that (4.10) is not sufficient to guarantee that the spectral radius of $V_q^T M V_q$ is smaller than 1. Hence, the Picard iterations may diverge. This could be cured by improving the basis, but we did not yet implement this in our code. In our experiments, we have never observed problems.

4.4 The extension to continuation.

In a continuation process, one of the bifurcation parameters, γ, is allowed to vary and a *parameterizing equation* (e.g. a *pseudo-arclength* equation [7]) is added to the BVP (2.2). This gives rise to an extra border column and border row in (3.5, 3.8). Analogously to [15, 16], the algorithm in Figure 4.1 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 iteration in the computation of the next periodic solution.

5 Examples.

We illustrate the convergence and efficiency of the continuation variant of the algorithm in Figure 4.1 by computing periodic solutions of two models.

Model A. In [20], 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 fixed 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]. By continuation, a branch of periodic solutions with bifurcation parameter $\beta_{2,1}$ was obtained. We computed an unstable periodic solution, denoted by A, with $\beta_{2,1} \approx 2.35001$ and $T \approx 66.3164$. The period of A is large because the branch approaches a heteroclinic orbit. The orbit of A is shown in Figure 5.1 (left).

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

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

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 λ. By continuation, 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 Figure 5.1 (right).

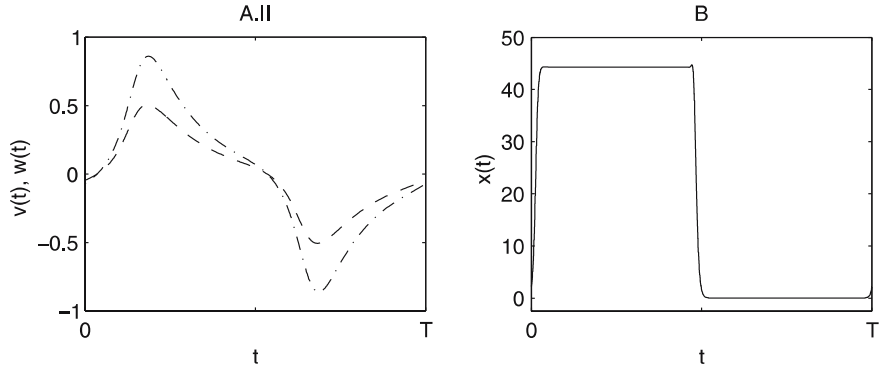

Figure 5.1: Left: the orbit of periodic solution A with v (dashed line) and w (dash–dotted line). Right: The orbit $x(t)$ of periodic solution B.

The periodic solutions were computed with a tolerance of RTOL $= 10^{-10}$ for the relative size of the residuals, i.e., (4.7). 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 5.1 lists the mesh characteristics m, N_t, ℓ and N. Note that as T and τ_k change during the iterations, ℓ may change and thus the dimension of u_s and u_f. However, this did not happen while computing these three periodic solutions. For periodic solution B, we needed a lot of mesh intervals to capture characteristics of the orbit such as the steep gradients and the little peek at $t \approx 0.5$ in Figure 5.1 (right).

Table 5.1: The values of the mesh characteristics m, N_t, ℓ and N.

	m	N_t	ℓ	N
A	144	864	3	20
B	1024	3072	1160	3481

Table 5.2 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 correction—as well as refined eigenvalues obtained by performing extra subspace iterations after computing the periodic solution until $\|QMV_p\|_2 \leqslant 10^{-5}$. The underlined digits correspond to those of 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.

Table 5.2: Left: The moduli of the dominant Floquet multipliers. For each periodic solution, the first row shows the Floquet multipliers as obtained in the last Newton step, while the second row shows these multipliers after performing additional subspace iterations at the end until $\|QMV_p\|_2 \leqslant 10^{-5}$. The digits of these refined Floquet multipliers that correspond to the plain DDE-BIFTOOL result are underlined. Right: p, after the last subspace iteration.

	$\lvert\mu_1\rvert$	$\lvert\mu_2\rvert$	p
A	5.694530 5.694558	1.000113 1.000108	2
B	1.008251 0.999999		1

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 Figure 5.2. The residual r_2 of the periodicity condition is zero before the first Newton iteration, since the initial orbit in the algorithm in Figure 4.1 is periodic. Remark that r_2 vanishes after the last Newton iteration in case of periodic solution A (see Figure 5.2, left) due to *numerical underflow*. For 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_{NP} is small enough to guarantee quadratic convergence of the Newton iteration.

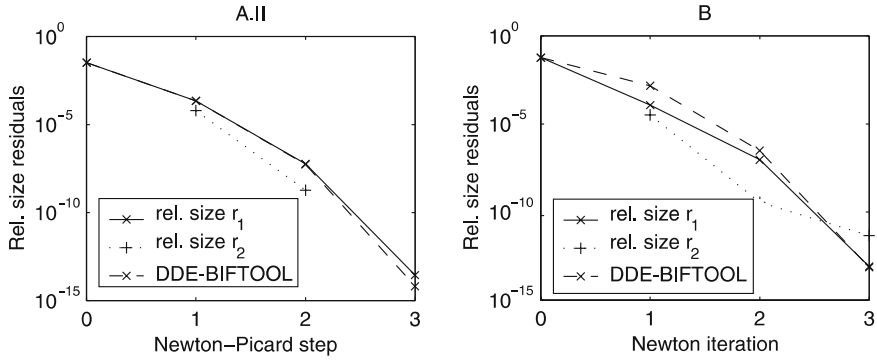

Figure 5.2: The relative size of the residuals vs. the Newton iteration for periodic solutions A (left) and B (right). For the Newton-Picard collocation method: the relative size of r_1 (×, solid line) and r_2 (+, dotted line) in (2.9a) and (2.9b), respectively. For DDE-BIFTOOL: the relative size of its collocation residual (×, dashed line).

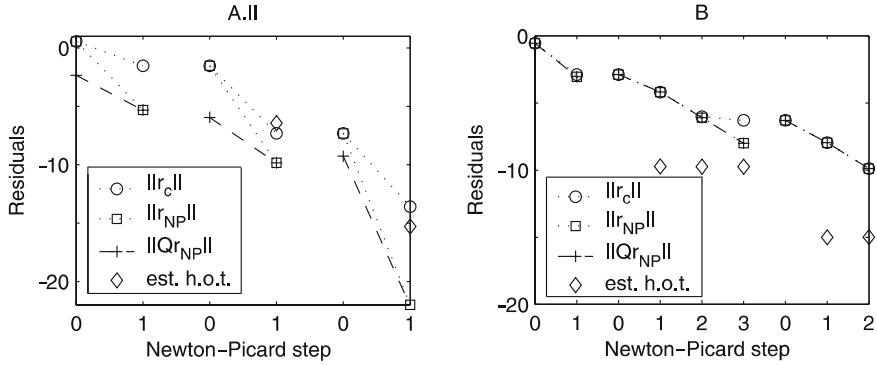

Figure 5.3: For periodic solutions A (left) and B (right): $\|r_c\|_2$ (○, dotted line), $\|r_{\mathrm{NP}}\|_2$ (□, dotted line), $\|Qr_{\mathrm{NP}}\|_2$ (+, dashed line) and an estimation of the higher-order terms (◇) vs. the Newton-Picard step.

Figure 5.3 illustrates the convergence of the Newton-Picard steps. 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 (3.6–3.7), though this value is only required at the start of each Newton iteration. We also show r_{NP} (the residual of (3.5, 3.8)) and its Q projection. Clearly, $r_{\mathrm{NP}} = r_c$ at the start of each Newton iteration and $\|r_{\mathrm{NP}}\|_2 \approx \|Qr_{\mathrm{NP}}\|_2$ after the the first Newton-Picard step. Furthermore, we also show the estimate (4.8) 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. The figures show that this estimate is rather crude. However, as is confirmed by Figure 5.2, it is good enough to preserve the quadratic convergence of the Newton iterations as is discussed in Section 4.3. Remark that the improvement in the last Newton-Picard step for example A is much larger than expected. Doing multiple Newton-Picard steps at each Newton iteration was clearly useful in these cases.

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 and B, 25 and 83 matrix-vector products with M were required, respectively.

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 5.3 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 con-

Table 5.3: 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").

	D.-B. P	NP-c P	D.-B. P+F	NP-c P+F
A	23.8	13.1	$23.8 + 4.9 = 28.7$	$13.1 + 0.3 = 13.4$
B	610.5	126.4	$610.5 + 689.1 = 1299.6$	$126.4 + 8.9 = 135.3$

tain 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 $\|QMV_p\|_2 \leqslant 10^{-5}$. In all these cases, our Newton-Picard collocation algorithm outperforms DDE-BIFTOOL. The difference is most significant for periodic solution B, where a fine discretization was needed. Indeed, the computation time in DDE-BIFTOOL is of order m^3 contrary to $\mathcal{O}(m)$ in our algorithm. Therefore, our method has clear advantages for larger values of m.

This argument also holds for the computation of the (dominant) Floquet multipliers, when DDE-BIFTOOL currently uses an algorithm with $\mathcal{O}(\max(\ell^3, m^3))$ complexity. Therefore our algorithm would also be interesting if τ/T is large.

6 Conclusions.

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

First, we have shown that the linearized collocation system can be solved robustly by using a condensed linear system. The resulting collocation procedure is equivalent to Newton iterations at the continuous level combined with single shooting for the resulting linear BVPs. This approach combines the advantages of collocation and single shooting methods without most of the disadvantages of either method. Remark that this approach can be generalized from collocation to many other discretization schemes.

Second, we have shown that the iterative Newton-Picard method is well suited to solve the condensed linear systems, especially when a fine discretization is used or for large systems of DDEs. We have adapted the method to the particular form of the nonlinear system and made various other improvements, the most important one the use of 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 iterations, contrary to previous implementations. Other applications of the Newton-Picard method (e.g., for parabolic PDEs) can also benefit from these improvements.

Remark that the Newton-Picard method is particularly attractive for bifurcation analysis because it produces good estimates for the dominant Floquet multipliers as a by-product. Moreover, we expect that the Newton-Picard collocation method can be extended (like its single shooting variant [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, 1995.
5. E. Doedel, A. Champneys, T. Fairgrieve, Y. Kuznetsov, B. Sandstede, 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. Engelborghs, *Numerical Bifurcation Analysis of Delay Differential Equations*, PhD thesis, Department of Computer Science, K. U. Leuven, Leuven, Belgium, May 2000.
9. K. Engelborghs 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. Engelborghs, 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. Engelborghs, 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. Engelborghs, 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. Engelborghs, T. Luzyanina, and G. Samaey, *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, 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. Engelborghs, *Computing Floquet multipliers for functional differential equations*, Int. J. Bifurcation Chaos, 11 (2002), pp. 737–753.

18. T. Luzyanina, K. Engelborghs, 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. Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Manchester University Press, 1992.

20. 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.

21. R. Weiss, *The application of implicit Runge-Kutta and collocation methods to boundary value problems*, Math. Comp., 28 (1974), pp. 449–464.