COMPUTATION AND BIFURCATION ANALYSIS OF PERIODIC SOLUTIONS OF LARGE-SCALE SYSTEMS

KURT LUST* AND DIRK ROOSE†

Abstract. This paper deals with the efficient computation and bifurcation analysis of periodic solutions of large-scale dynamical systems, such as systems arising from the spatial discretization of partial differential equations. The Newton–Picard method is an efficient single-shooting based technique based on a Newton-like linearization which exploits the low-dimensional dynamics observed in many systems. The dominant, stability-determining Floquet multipliers are easily recovered from the computations. In this paper, we develop an algebraic framework which generalizes older variants of the Newton–Picard method (including the Recursive Projection method) and which allows us to explain and to monitor the convergence behavior. Special attention is paid to algorithmic aspects which improve the robustness of the method. The efficiency of the approach is illustrated by some numerical results.

Key words. dynamical systems, periodic solutions, shooting, bifurcation analysis, stability

AMS(MOS) subject classifications. 65N12, 35B10, 35B32

1. Introduction.

1.1. Continuation of periodic solutions. This paper deals with the efficient computation of periodic solutions of partial differential equations (PDEs) and the determination of their stability. We consider a parameter-dependent autonomous dynamical system

$$\frac{dx}{dt} = f(x,\gamma),\ x \in \mathbb{R}^N,\ \gamma \in \mathbb{R}, \tag{1.1}$$

with N "large", and with f derived from a finite element or finite difference spatial discretization of a parabolic PDE. We will assume that f is C^2-continuous in x and γ in the region of interest. For a fixed value of the parameter γ, a periodic solution is determined by $N+1$ unknowns, namely, the initial conditions $x(0) \in \mathbb{R}^N$ and the period T. To find these unknowns we use the system

$$\begin{cases} x(T) - x(0) = 0, \\ s(x(0), T) = 0, \end{cases}$$

where the second equation is a phase condition needed to eliminate the invariance of periodic solutions of autonomous dynamical systems under time translation (see, e.g., [23]). When computing a branch of periodic

*IMA, University of Minnesota, Vincent Hall 514, 206 Church Street S.E., Minneapolis MN 55455, U.S.A. E-mail: Kurt.Lust@cs.kuleuven.ac.be.

†Department of Computer Science, K.U.Leuven, Celestijnenlaan 200A, B-3001 Heverlee, Belgium. E-mail: Dirk.Roose@cs.kuleuven.ac.be.

solutions, the parameter γ is allowed to vary and an additional condition, the parameterizing equation, is added. The unknowns $x(0)$, T and γ are found by solving

$$\begin{cases} x(T) - x(0) = 0, \\ s(x(0), T, \gamma) = 0, \\ n(x(0), T, \gamma; \eta) = 0, \end{cases} \tag{1.2}$$

with η an artificial parameter. With a good choice for n, one is able to pass around fold points and to detect other bifurcations without any difficulties. We use pseudo-arclength parameterization [2], but other choices are possible [19, 23].

1.2. Stability of periodic solutions. Let $\varphi(x(0), T, \gamma)$ denote the solution of (1.1) at $t = T$ for a given initial condition $x(0)$ and let $(x(0)^*, T^*, \gamma^*)$ denote a solution of (1.2). If $\frac{\partial \varphi}{\partial x}$ denotes the partial derivative of $\varphi(x(0), T, \gamma)$ with respect to $x(0)$, then the monodromy matrix M^* is given by

$$M^* := \left. \frac{\partial \varphi}{\partial x} \right|_{(x^*(0), T^*, \gamma^*)}.$$

M^* is a full and nonnormal matrix. We denote its eigenvalues, i.e., the *Floquet multipliers,* by μ_i^*, $i = 1, \ldots, N$. As is well known [23], one eigenvalue of M^* has the value 1. The corresponding eigenvector is the tangent vector to the limit cycle in $(x(0)^*, \gamma^*)$,

$$b_T^* := \left. \frac{\partial \varphi}{\partial T} \right|_{(x(0)^*, T^*, \gamma^*)} = f(\varphi(x(0)^*, T^*, \gamma^*), \gamma^*).$$

If all Floquet multipliers except the trivial Floquet multiplier 1 lie inside the unit circle, the periodic orbit is (asymptotically) stable. If at least one Floquet multiplier lies outside the unit circle, the periodic orbit is unstable.

If f in (1.1) is the result of a space discretization of a parabolic PDE, there are usually few Floquet multipliers close to or outside the unit circle. Moreover, the values and the number of those Floquet multipliers are basically independent of the discretization used and the dimension N of the discretized system. This property will be exploited in the numerical method proposed in this paper and this will lead to a substantial reduction in computational cost, compared with classical approaches to compute periodic solutions and their stability.

1.3. Computation of periodic solutions. The method described in this paper is based on single shooting (the extension to multiple shooting is discussed in [14]). In this approach, the nonlinear system

$$\begin{cases} r(x(0), T, \gamma) & := & \varphi(x(0), T, \gamma) - x(0) = 0, \\ s(x(0), T, \gamma) & = & 0, \\ n(x(0), T, \gamma; \eta) & = & 0 \end{cases} \tag{1.3}$$

(equivalent to the boundary value problem (1.1)–(1.2)) is solved for the unknowns $x(0)$, T and γ, using a Newton-like method. If a direct linear system solver is used to solve the linearized problem, the $N \times N$-matrix $\frac{\partial \varphi}{\partial x}$ must be computed or approximated by solving the variational form of (1.1) or by numerical differentiation. In the former case, an N^2-dimensional linear initial-value problem must be integrated together with (1.1), and in the latter case the nonlinear system (1.1) must be integrated N times with perturbed initial data. Both approaches are prohibitively expensive when N is large. Also the storage and factorization of the full $N \times N$-matrix is expensive.

In this paper, we develop an efficient single-shooting based technique, based on a Newton-like linearization, in which the explicit calculation of the matrix $\frac{\partial \varphi}{\partial x}$ is avoided. This approach requires only the calculation of the action of $\frac{\partial \varphi}{\partial x}$ on a p-dimensional subspace of $\mathbb{R}^N$ with $p \ll N$, a computationally less expensive task. The linearized system is solved by using a combination of direct linear system solvers and iterative solvers (e.g., Picard iteration). We also recover information on the dominant, stability-determining Floquet multipliers. The proposed method is an extension of the Newton–Picard method presented in [15, 20] and is also related to the Recursive Projection Method (RPM) of Shroff and Keller [24] and the condensed Newton–supported Picard method of Jarausch and Mackens [9, 10, 11]. Compared with the approach in [15, 20], we develop in this paper a general algebraic framework, which covers the methods presented in [15] and which allows us to explain and to monitor their convergence behavior.

Note that our approach is influenced by the knowledge that the problem is set in a continuation framework, and hence, for a given γ, reasonably accurate starting approximations for $x(0)$ and T and for the Floquet multipliers are known. This makes some of the linear algebra techniques much more robust than when used in a one-off problem.

The plan of the paper is as follows. In section 2 we derive the algebraic framework from which several variants of the Newton–Picard method can be derived and we present a strategy for analyzing the convergence behavior. In section 3 we study the convergence behavior of several Newton–Picard variants. In section 4 we show how to use the convergence analysis to develop a very robust method. The computation of the basis for the low-dimensional subspace needed in the algorithm is discussed in section 5 and the solution of the two types of linear systems that occur is discussed in section 6 and 7. In section 8 we present some test results. Finally, section 9 presents our conclusions.

2. The Newton–Picard method: algebraic framework.

2.1. Derivation of the method. A single shooting pseudo-arclength continuation method solves the nonlinear system (1.3) for the unknowns

$x(0)$, T and γ. We assume that φ, s and n are twice differentiable with respect to $x(0)$, T and γ. For ease of implementation, the parameterizing equation $n(x(0), T, \gamma; \eta) = 0$ should not involve time integration or integrals along the limit cycle. In our experiments, we used the pseudo-arclength parameterization

$$\begin{aligned} n(x(0), T, \gamma; \eta) &= \theta_x \left(x(0) - x_p(0)\right)^T x_s(0) + \theta_T \left(T - T_p\right) T_s \\ &\quad + \theta_\gamma \left(\gamma - \gamma_p\right) \gamma_s = 0, \end{aligned} \tag{2.1}$$

where $(x_p(0), T_p, \gamma_p)$ is the point generated by the predictor and $(x_s(0), T_s, \gamma_s)$ is the normalized predictor direction. This equation forces the solution to lie in the hypersurface through the predicted point and orthogonal to the predictor direction in some weighted scalar product. We refer to [2, 23] for other parameterizing equations.

The use of Newton's method for solving (1.3) leads to the repetitive solution of linear systems of the form

$$\begin{aligned} &\begin{bmatrix} M^{(\nu)} - I & b_T^{(\nu)} & b_\gamma^{(\nu)} \\ {c_s^{(\nu)}}^T & d_{s,T}^{(\nu)} & d_{s,\gamma}^{(\nu)} \\ {c_n^{(\nu)}}^T & d_{n,T}^{(\nu)} & d_{n,\gamma}^{(\nu)} \end{bmatrix} \begin{bmatrix} \Delta x(0)^{(\nu)} \\ \Delta T^{(\nu)} \\ \Delta \gamma^{(\nu)} \end{bmatrix} \\ &= - \begin{bmatrix} r(x(0)^{(\nu)}, T^{(\nu)}, \gamma^{(\nu)}) \\ s(x(0)^{(\nu)}, T^{(\nu)}, \gamma^{(\nu)}) \\ n(x(0)^{(\nu)}, T^{(\nu)}, \gamma^{(\nu)}; \eta) \end{bmatrix}, \end{aligned} \tag{2.2}$$

where

$$\begin{bmatrix} M^{(\nu)} - I & b_T^{(\nu)} & b_\gamma^{(\nu)} \\ {c_s^{(\nu)}}^T & d_{s,T}^{(\nu)} & d_{s,\gamma}^{(\nu)} \\ {c_n^{(\nu)}}^T & d_{n,T}^{(\nu)} & d_{n,\gamma}^{(\nu)} \end{bmatrix} = \left. \frac{\partial(\varphi, s, n)(x(0), T, \gamma)}{\partial(x(0), T, \gamma)} \right| (x(0)^{(\nu)}, T^{(\nu)}, \gamma^{(\nu)}) \ ,$$

followed by an update

$$\begin{aligned} x(0)^{(\nu+1)} &= x(0)^{(\nu)} + \Delta x(0)^{(\nu)}, \\ T^{(\nu+1)} &= T^{(\nu)} + \Delta T^{(\nu)}, \\ \gamma^{(\nu+1)} &= \gamma^{(\nu)} + \Delta \gamma^{(\nu)}. \end{aligned}$$

From now on, we drop the superscript (ν) from our notation whenever the formula remains clear. With an appropriate parameterizing equation, (2.2) will be nonsingular at hyperbolic periodic solutions, fold points and all bifurcation points that do not involve an (additional) $+1$ Floquet multiplier.

We make the following assumptions.

ASSUMPTION 2.1. *Let $y^* = (x(0)^*, T^*, \gamma^*)$ denote an isolated solution to (1.3), and let $\mathcal{B}$ be a small neighborhood of y^*. Let $M(y) = \frac{\partial \varphi}{\partial x(0)}(y)$ for $y \in \mathcal{B}$ and denote its eigenvalues by μ_i, $i = 1, \ldots, N$. Assume that for all $y \in \mathcal{B}$ precisely p eigenvalues lie outside the disk*

$$C_\rho = \{|z| < \rho\}, \quad 0 < \rho < 1 \tag{2.3}$$

and that no eigenvalue has modulus ρ; i.e., for all $y \in \mathcal{B}$

$$|\mu_1| \geq |\mu_2| \geq \cdots \geq |\mu_p| > \rho > |\mu_{p+1}|, \ldots, |\mu_N|.$$

Note that $M(y^*) = M^*$, i.e., the monodromy matrix. Our method is designed to be efficient for $p \ll N$. Let the column vectors of $V_p \in \mathbb{R}^{N\times p}$ define an orthonormal basis for the subspace $\mathcal{U}$ of $\mathbb{R}^N$ spanned by the (generalized) eigenvectors of $M(x(0), T, \gamma)$ corresponding to the eigenvalues μ_i, $i = 1, \ldots, p$, and the column vectors of $V_q \in \mathbb{R}^{N\times(N-p)} = \mathbb{R}^{N\times q}$ define an orthonormal basis for $\mathcal{U}^\perp$, the orthogonal complement of $\mathcal{U}$ in $\mathbb{R}^N$. In general, $\mathcal{U}^\perp$ is not an invariant subspace of $M(x(0), T, \gamma)$. V_p and V_q could be computed by the real Schur factorization of $M(x(0), T, \gamma)$. However, we will use V_q, the basis for the high-dimensional space $\mathcal{U}^\perp$, only in the derivation of the method, but not in the final algorithms (see section 6). In actual computations eigenvalues may move out of or into C_ρ as the iteration proceeds and our code is able to deal with this. However as the iteration comes close to convergence, the approximate solutions y remain in $\mathcal{B}$ and Assumption 2.1 holds.

The orthogonal projectors P and Q of $\mathbb{R}^N$ onto $\mathcal{U}$ and $\mathcal{U}^\perp$ are given by

$$\begin{aligned} P &:= V_p V_p^T, \\ Q &:= V_q V_q^T = I_N - V_p V_p^T. \end{aligned} \tag{2.4}$$

Qv can be computed without actually knowing the matrix V_q. For any $\Delta x(0) \in \mathbb{R}^N$ there is the unique decomposition

$$\begin{aligned} \Delta x(0) &= V_p \Delta\bar{p} + V_q \Delta\bar{q} = \Delta p + \Delta q, \\ \Delta p &= V_p \Delta\bar{p} = P\Delta x(0), \\ \Delta q &= V_q \Delta\bar{q} = Q\Delta x(0) \end{aligned} \tag{2.5}$$

with $\Delta\bar{p} \in \mathbb{R}^p$ and $\Delta\bar{q} \in \mathbb{R}^{N-p}$. Substituting (2.5) in (2.2) and multiplying the first N equations at the left-hand side with $[V_q \;\; V_p]^T$, one obtains

$$\begin{aligned} &\begin{bmatrix} V_q^T M V_q - I_q & 0 & V_q^T b_T & V_q^T b_\gamma \\ V_p^T M V_q & V_p^T M V_p - I_p & V_p^T b_T & V_p^T b_\gamma \\ c_s^T V_q & c_s^T V_p & d_{s,T} & d_{s,\gamma} \\ c_n^T V_q & c_n^T V_p & d_{n,T} & d_{n,\gamma} \end{bmatrix} \begin{bmatrix} \Delta\bar{q} \\ \Delta\bar{p} \\ \Delta T \\ \Delta\gamma \end{bmatrix} \\ &\qquad = - \begin{bmatrix} V_q^T r \\ V_p^T r \\ s \\ n \end{bmatrix}. \end{aligned} \tag{2.6}$$

Note that since V_p and V_q define bases for orthogonal subspaces and since V_p is a basis for an invariant subspace of M, $V_q^T M V_p$ is zero. At the limit

cycle, the vector b_T is an eigenvector of M for the trivial Floquet multiplier 1 and thus a vector in the subspace $\mathcal{U}$. Hence, during the Newton iteration the term $V_q^T b_T$ is usually small and was neglected in the methods presented in [15].

System (2.6) can be solved by first applying one step of block-Gaussian elimination to reduce (2.6) to an upper block triangular system. This corresponds to first computing $\Delta\bar{q}_r$, $\Delta\bar{q}_T$ and $\Delta\bar{q}_\gamma$ by solving the $(N-p)$-dimensional systems

$$\begin{aligned}\left[V_q^T M V_q - I_q\right]&\left[\begin{array}{ccc}\Delta\bar{q}_r & \Delta\bar{q}_T & \Delta\bar{q}_\gamma\end{array}\right] = \\ &-\left[\begin{array}{ccc} V_q^T r & V_q^T b_T & V_q^T b_\gamma\end{array}\right],\end{aligned} \tag{2.7}$$

then computing $\Delta\bar{p}$, ΔT and $\Delta\gamma$ by solving the $(p+2)$-dimensional system

$$\begin{aligned}&\left[\begin{array}{ccc} V_p^T M V_p - I_p & V_p^T(b_T + M V_q \Delta\bar{q}_T) & V_p^T(b_\gamma + M V_q \Delta\bar{q}_\gamma)\\ c_s^T V_p & d_{s,T} + c_s^T V_q \Delta\bar{q}_T & d_{s,\gamma} + c_s^T V_q \Delta\bar{q}_\gamma\\ c_n^T V_p & d_{n,T} + c_n^T V_q \Delta\bar{q}_T & d_{n,\gamma} + c_n^T V_q \Delta\bar{q}_\gamma\end{array}\right]\\ &\left[\begin{array}{c}\Delta\bar{p}\\ \Delta T\\ \Delta\gamma\end{array}\right] = -\left[\begin{array}{c} V_p^T(r + M V_q \Delta\bar{q}_r)\\ s + c_s^T V_q \Delta\bar{q}_r\\ n + c_n^T V_q \Delta\bar{q}_r\end{array}\right]\end{aligned} \tag{2.8}$$

and finally computing

$$\Delta\bar{q} = \Delta\bar{q}_r + \Delta T\,\Delta\bar{q}_T + \Delta\gamma\,\Delta\bar{q}_\gamma. \tag{2.9}$$

We call the three systems contained in (2.7) the Q-systems and the system (2.8) the P-system. Note that the matrix in (2.8) is nonsingular iff the matrix in (2.2) is nonsingular. All information about stability and the occurrence of bifurcations is preserved in the low-dimensional system (2.8).

At this point, we have not yet obtained any savings. In fact, the above scheme is even more expensive than using Gaussian elimination to solve (2.2). To reduce the computational cost significantly, we make the following approximations:

- The basis V_p is not computed exactly. Instead, an approximate basis is computed using orthogonal subspace iteration. We will discuss this procedure in section 5. Note that the term $V_q^T M V_p$ is no longer zero and a nonzero element should be introduced at the $(1,2)$-position in (2.6). However, we will neglect this term.
- The Q-systems (2.7) are solved approximately using iterative methods, e.g., Picard iteration. To avoid the need for the basis V_q, we do not compute the $(N-p)$-dimensional vectors $\Delta\bar{q}_r$, $\Delta\bar{q}_T$ and $\Delta\bar{q}_\gamma$, but we compute directly the N-dimensional vectors $\Delta q_r = V_q\Delta\bar{q}_r$, $\Delta q_T = V_q\Delta\bar{q}_T$ and $\Delta q_\gamma = V_q\Delta\bar{q}_\gamma$ (see section 6). Note however that the Q-systems can also be solved by more powerful iterative methods (e.g., GMRES), as suggested in [14].

Since we use a Picard iteration to solve the Q-systems, we have called our method the Newton–Picard method. The variant described above is

denoted by the abbreviation CNP, for continuation variant of the Newton–Picard method.

In the fixed parameter case, i.e., when γ is considered to be fixed, we can omit the last row and column in (2.8) and the term in $\Delta\bar{q}_\gamma$ in (2.7) and (2.9). We first compute $\Delta\bar{q}_r$ and $\Delta\bar{q}_T$ from

$$[V_q^T M V_q - I_q] \left[\begin{array}{cc} \Delta\bar{q}_r & \Delta\bar{q}_T \end{array} \right] = - \left[\begin{array}{cc} V_q^T r & V_q^T b_T \end{array} \right], \tag{2.10}$$

using a Picard iteration, then compute $\Delta\bar{p}$ and ΔT from

$$\begin{array}{l} \left[\begin{array}{cc} V_p^T M V_p - I_p & V_p^T (b_T + M V_q \Delta\bar{q}_T) \\ c_s^T V_p & d_{s,T} + c_s^T V_q \Delta\bar{q}_T \end{array} \right] \left[\begin{array}{c} \Delta\bar{p} \\ \Delta T \end{array} \right] = \\ \qquad - \left[\begin{array}{c} V_p^T (r + M V_q \Delta\bar{q}_r) \\ s + c_s^T V_q \Delta\bar{q}_r \end{array} \right], \end{array} \tag{2.11}$$

and finally compute

$$\Delta\bar{q} = \Delta\bar{q}_r + \Delta T \, \Delta\bar{q}_T. \tag{2.12}$$

This variant is denoted by the abbreviation $NPGS$, because of the similarity with the method that we developed in [15] and denoted by $NPGS(l)$ (Newton–Picard Gauss–Seidel method).

Since $M = \partial\varphi(x(0), T, \gamma)/\partial x(0)$, the matrix-vector products Mv are rescaled directional derivatives. They can be computed without requiring the full monodromy matrix M, using a variational equation or finite difference approximation. If a variational equation is used, $Mv = z(T)$ with $z(t)$ the solution of the initial value problem

$$\dot{z} = f_x(\varphi(x(0), t, \gamma), \gamma) z, \; z(0) = v.$$

This N-dimensional linear differential equation must be integrated together with the nonlinear equation (1.1) if the trajectory $\varphi(x(0), t, \gamma)$ has not been stored. The second approach is to approximate Mv using a finite difference formula, e.g., the first-order formula

$$Mv \approx \frac{1}{\epsilon} \left[\varphi(x(0) + \epsilon v, T, \gamma) - \varphi(x(0), T, \gamma) \right].$$

This requires one additional time integration of the N-dimensional nonlinear equation (1.1). Note that both time integrations should be done using the same sequence of time steps and order of the method when using a variable stepsize/order method, or the time integrations must be computed with very high accuracy. More details on both procedures can be found in [15, 14].

2.2. Convergence analysis. We now derive the relationship between the error after a Newton–Picard step on one hand, and the residuals of the Q-systems (2.7), the accuracy of the basis and the higher-order terms on the other hand. This relationship not only explains why some of the variants developed in [15] sometimes perform badly or even fail, but it can also be used to control the convergence of the method described above. Based on this convergence analysis we will develop more robust variants of the Newton–Picard method in section 4.

Let

$$\begin{bmatrix} Qr_r & Qr_T & Qr_\gamma \end{bmatrix} = \begin{bmatrix} Qr & Qb_T & Qb_\gamma \end{bmatrix} + (QMQ - I) \begin{bmatrix} \Delta q_r & \Delta q_T & \Delta q_\gamma \end{bmatrix} \tag{2.13}$$

(with $\Delta q_r = V_q \Delta \bar{q}_r$, etc.) be the N-dimensional residuals of the Q-systems (2.7) (i.e., $V_q \times$ the actual residuals of (2.7)). We will use the notation Δq_* to denote the three vectors $\Delta q_r, \Delta q_T$ and Δq_γ. From a Taylor expansion around the starting point $(x(0), T, \gamma)$ of the Newton-Picard step we get

$$\begin{aligned} &Qr(x(0) + \Delta q + \Delta p, T + \Delta T, \gamma + \Delta\gamma) = \\ &\quad Qr(x(0), T, \gamma) + QM\Delta q - \Delta q + QM\Delta p \\ &\quad + \Delta T\, Qb_T + \Delta\gamma\, Qb_\gamma + \mathrm{O}(\Delta q, \Delta p, \Delta T, \Delta\gamma)^2. \end{aligned} \tag{2.14}$$

Using (2.9) (in terms of the N-dimensional vectors Δq_* instead of the $(N-p)$-dimensional vectors $\Delta \bar{q}_*$) and (2.13), this relation can be rewritten as

$$\begin{aligned} &Qr(x(0) + \Delta q + \Delta p, T + \Delta T, \gamma + \Delta\gamma) = Qr_r + \Delta T\, Qr_T \\ &\quad + \Delta\gamma\, Qr_\gamma + QM\Delta p + \mathrm{O}(\Delta q, \Delta p, \Delta T, \Delta\gamma)^2. \end{aligned} \tag{2.15}$$

Similarly,

$$\begin{cases} V_p^T r(x(0) + \Delta q + \Delta p, T + \Delta T, \gamma + \Delta\gamma) = \\ \quad V_p^T r(x(0), T, \gamma) + V_p^T M V_q \Delta\bar{q} + V_p^T M V_p \Delta\bar{p} - V_p^T \Delta\bar{p} \\ \quad + \Delta T\, V_p^T b_T + \Delta\gamma\, V_p^T b_\gamma + \mathrm{O}(\Delta q, \Delta p, \Delta T, \Delta\gamma)^2, \\ s(x(0) + \Delta q + \Delta p, T + \Delta T, \gamma + \Delta\gamma) = s(x(0), T, \gamma) + c_s^T V_q \Delta\bar{q} \\ \quad + c_s^T V_p \Delta\bar{p} + d_{s,T}\Delta T + d_{s,\gamma}\Delta\gamma + \mathrm{O}(\Delta q, \Delta p, \Delta T, \Delta\gamma)^2, \\ n(x(0) + \Delta q + \Delta p, T + \Delta T, \gamma + \Delta\gamma) = n(x(0), T, \gamma) + c_n^T V_q \Delta\bar{q} \\ \quad + c_n^T V_p \Delta\bar{p} + d_{n,T}\Delta T + d_{n,\gamma}\Delta\gamma + \mathrm{O}(\Delta q, \Delta p, \Delta T, \Delta\gamma)^2. \end{cases}$$

By first substituting (2.9) in the right-hand side of this expression, then comparing the resulting expression with (2.8) to eliminate most terms and finally premultiplying the first set of equations with V_p, one can show that

$$\begin{cases} Pr(x(0) + \Delta q + \Delta p, T + \Delta T, \gamma + \Delta\gamma) \\ \quad = \mathrm{O}(\Delta q, \Delta p, \Delta T, \Delta\gamma)^2, \\ s(x(0) + \Delta q + \Delta p, T + \Delta T, \gamma + \Delta\gamma) \\ \quad = \mathrm{O}(\Delta q, \Delta p, \Delta T, \Delta\gamma)^2, \\ n(x(0) + \Delta q + \Delta p, T + \Delta T, \gamma + \Delta\gamma) \\ \quad = \mathrm{O}(\Delta q, \Delta p, \Delta T, \Delta\gamma)^2. \end{cases} \tag{2.16}$$

Note that we supposed that the (low-dimensional) P-system (2.8) is solved accurately. Otherwise the residual

$$\begin{bmatrix} V_p^T(r + M\Delta q_r) \\ s + c_s^T \Delta q_r \\ n + c_n^T \Delta q_r \end{bmatrix} + \begin{bmatrix} V_p^T M V_p - I_p & V_p^T(b_T + M\Delta q_T) & V_p^T(b_\gamma + M\Delta q_\gamma) \\ c_s^T V_p & d_{s,T} + c_s^T \Delta q_T & d_{s,\gamma} + c_s^T \Delta q_\gamma \\ c_n^T V_p & d_{n,T} + c_n^T \Delta q_T & d_{n,\gamma} + c_n^T \Delta q_\gamma \end{bmatrix} \begin{bmatrix} \Delta\bar{p} \\ \Delta T \\ \Delta\gamma \end{bmatrix} \tag{2.17}$$

must be added in the right-hand side of (2.16).

The relations (2.15) and (2.16) allow us to analyze and to monitor the convergence of (variants of) the Newton–Picard method. At the end of the Newton–Picard step, all terms in these relations can be computed without extra matrix–vector products or time integrations of (1.1). The evaluation of $Qr(x(0) + \Delta q + \Delta p, T + \Delta T, \gamma + \Delta\gamma)$ and $Pr(x(0) + \Delta q + \Delta p, T + \Delta T, \gamma + \Delta\gamma)$ requires a time integration of (1.1), but we need to do that time integration anyway at the beginning of the next iteration step and as a test for convergence. The matrix–vector products $M\Delta q_*$ needed for the evaluation of the residuals (2.13) are also needed to build the P-system (2.8). From the subspace iteration to compute the basis V_p (see section 5) we can recover MV_p, so

$$QM\Delta p = \left(I - V_p V_p^T\right) MV_p \Delta\bar{p}$$

can also be computed easily. The evaluation of (2.17) is also cheap (if needed at all). Hence the higher-order terms in (2.15) and (2.16) can also be computed, since all other terms are known. Note that we assume here that the numerical errors made in the evaluation of each of the terms are much smaller than the size of each of those terms, i.e., we assume that all time integrations and matrix–vector products are computed accurately enough.

Let us now discuss the effect of some of the terms in (2.15) and (2.16) in more detail. This discussion is mathematically not very rigorous but it gives an intuitive explanation of the performance of the methods.

- The Picard iteration to solve the Q-systems (2.7) has a linear asymptotic convergence rate $|\mu_{p+1}|$. If we use starting values $\Delta q_* = 0$, we expect that after the Picard iteration $\|Qr_r\| < \|Qr(x(0), T, \gamma)\|$, $\|Qr_T\| < \|Qb_T\|$ and $\|Qr_\gamma\| < \|Qb_\gamma\|$ (since (2.14) yields $Qr_r = Qr(x(0), T, \gamma)$, etc., at the beginning of the Picard iteration). However, since M is a nonnormal matrix, the residuals (2.13) may initially grow before converging. This may cause divergence of the Newton–Picard step if the number of Picard iterations is fixed in advance. We have never observed this phenomenon in practice, but it is clear that a truly robust method should use an adaptive number of Picard iteration steps.

- The update $(\Delta\bar{p}, \Delta T, \Delta\gamma)$ does not only depend on the P-projection of the initial residual, but also on the Q-projection because of the terms $V_p^T M V_q$, $c_s^T V_q$ and $c_n^T V_q$ in (2.8). In general, the norms of Δq_r and $PM\Delta q_r$ are of the same order of magnitude as the norm of $Qr(x(0), T, \gamma)$. If $Qr(x(0), T, \gamma)$ is much larger than $Pr(x(0), T, \gamma)$, the P-system (2.8) usually has a large right-hand side since Δq_r and thus $V_p^T M \Delta q_r$ (and possibly, $c_s^T \Delta q_r$ and $c_n^T \Delta q_r$) are large. The norm of the right-hand side of the P-system is typically of the same order of magnitude as the maximum of the norms of the P- and Q-projection of the initial residual and the update $(\Delta\bar{p}, \Delta T, \Delta\gamma)$ will be large if either $Pr(x(0), T, \gamma)$ or $Qr(x(0), T, \gamma)$ are large. The size of the terms $\Delta T\, Qb_T$, $\Delta\gamma\, Qb_\gamma$ and $QM\Delta p$ thus depends on both $Pr(x(0), T, \gamma)$ and $Qr(x(0), T, \gamma)$, and will be large if either $Pr(x(0), T, \gamma)$ or $Qr(x(0), T, \gamma)$ are large.
- As discussed before, Qb_T is often small. Hence we can often neglect this term (as we did in [15]) and set $\Delta q_T = 0$. However, we have observed that setting $\Delta q_T = 0$ sometimes slows down the convergence during the first Newton–Picard steps.
- The term Qb_γ on the other hand may be quite large and setting $\Delta q_\gamma = 0$ (hence $Qr_\gamma = Qb_\gamma$) may cause convergence problems. The right-hand side of the P-system (2.8) is usually at least of the order of magnitude of the Q-projection of the initial residual. Hence $\Delta\gamma$ and $\Delta\gamma\, Qb_\gamma$ can be quite large. Thus a robust method for continuation should not set $\Delta q_\gamma = 0$ without first testing for the size of Qb_γ.

3. Convergence analysis of some Newton–Picard variants.

In [15], we developed four variants of the method described above, two for a fixed parameter and two for pseudo-arclength continuation. All variants were derived by approximating terms in (2.6):

- the term $V_q^T b_T$ is neglected since this term is usually small,
- the term $V_q^T M V_q - I_q$ is approximated by

$$-\left(\sum_{i=0}^{l-1} \left(V_q^T M V_q\right)^i\right)^{-1}. \tag{3.1}$$

In some variants, we also neglected other terms. We now describe these variants in more detail.

3.1. Fixed parameter variants *NPGS(l)* and *NPJ(l)*. When the parameter γ is considered to be fixed, the last row and column and the terms $\Delta\gamma$ and n in (2.6) can be omitted, leading to the systems (2.10) and (2.11).

NPGS(l). A first fixed parameter variant is obtained by using the approximation (3.1) for $V_q^T M V_q - I_q$ and neglecting $V_q^T b_T$ in (2.10). Hence

$\Delta\bar{q}_T = 0$ and

$$\Delta q = V_q \Delta\bar{q} = V_q \sum_{i=0}^{l-1} \left(V_q^T M V_q\right)^i V_q^T r, \tag{3.2}$$

which corresponds to l Picard iterations with starting value $\Delta\bar{q} = 0$. $\Delta\bar{p}$ and ΔT are solved from (2.10) which reduces to

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T b_T \\ c_s^T V_p & d_{s,T} \end{bmatrix} \begin{bmatrix} \Delta\bar{p} \\ \Delta T \end{bmatrix} = - \begin{bmatrix} V_p^T (r + M\Delta q) \\ s + c_s^T \Delta q \end{bmatrix}. \tag{3.3}$$

The first term in the right-hand side of this system is up to higher order terms equal to $V_p^T r(x(0) + \Delta q, T, \gamma)$, i.e., the P-projection of the residual at the end of the Picard iterations. If $l = 1$, this scheme corresponds to first doing a time integration Picard iteration step restricted to the space $\mathcal{U}^\perp$ followed by a Newton-based single shooting step in the space $\mathcal{U}$ starting from the end point of the time integration. Therefore, we have called this scheme the Newton–Picard Gauss–Seidel method, denoted as $NPGS(l)$, where l stands for the number of Picard iteration steps.

NPJ(l). A second fixed parameter variant is derived by also neglecting the terms $V_p^T M V_q$ and $c_s^T V_q$ in (2.6) and (2.11). Δq is computed using (3.2) and $\Delta\bar{p}$ and ΔT are solved from

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T b_T \\ c_s^T V_p & d_{s,T} \end{bmatrix} \begin{bmatrix} \Delta\bar{p} \\ \Delta T \end{bmatrix} = - \begin{bmatrix} V_p^T r \\ s \end{bmatrix}. \tag{3.4}$$

Because (3.2) and (3.4) are fully decoupled, we have called this variant the Newton–Picard Jacobi method with l Picard steps, denoted as $NPJ(l)$. Note that the $NPJ(1)$ method corresponds to the fixed parameter variant of the recursive projection method developed by Shroff and Keller [24].

3.2. Continuation variants *CNP(l)* and *CRP(l)*.

CNP(l). A first continuation variant is obtained by using the approximation for $V_q^T M V_q - I_q$ and neglecting $V_q^T b_T$ in (2.6), i.e., we solve the linear system

$$\begin{bmatrix} -\left(\sum_{i=0}^{l-1} \left(V_q^T M V_q\right)^i\right)^{-1} & 0 & 0 & V_q^T b_\gamma \\ V_p^T M V_q & V_p^T M V_p - I_p & V_p^T b_T & V_p^T b_\gamma \\ c_s^T V_q & c_s^T V_p & d_{s,T} & d_{s,\gamma} \\ c_n^T V_q & c_n^T V_p & d_{n,T} & d_{n,\gamma} \end{bmatrix} \begin{bmatrix} \Delta\bar{q} \\ \Delta\bar{p} \\ \Delta T \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} V_q^T r \\ V_p^T r \\ s \\ n \end{bmatrix}$$

exactly. In [15], we propose to use the Sherman–Morrison formula to solve this system. Alternatively, we can use the block Gauss elimination method proposed above. In this case, we first compute Δq_r and Δq_γ from

$$\Delta q_r = V_q \sum_{i=0}^{l-1} \left(V_q^T M V_q\right)^i V_q^T r,$$
$$\Delta q_\gamma = V_q \sum_{i=0}^{l-1} \left(V_q^T M V_q\right)^i V_q^T b_\gamma,$$

then $\Delta\bar{p}$, ΔT and $\Delta\gamma$ are computed from

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T b_T & V_p^T\left(b_\gamma + M\Delta q_\gamma\right) \\ c_s^T V_p & d_{s,T} & d_{s,\gamma} + c_s^T \Delta q_\gamma \\ c_n^T V_p & d_{n,T} & d_{n,\gamma} + c_n^T \Delta q_\gamma \end{bmatrix} \begin{bmatrix} \Delta\bar{p} \\ \Delta T \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} V_p^T\left(r + M\Delta q_r\right) \\ s + c_s^T \Delta q_r \\ n + c_n^T \Delta q_r \end{bmatrix}$$

and finally we compute

$$\Delta q = \Delta q_r + \Delta\gamma\, \Delta q_\gamma.$$

This variant is denoted as $CNP(l)$, for continuation variant of the Newton–Picard method with l Picard steps. Note that this method requires two sequences of l Picard iteration steps and is considerably more expensive than the fixed parameter methods if l is large.

CRP(l). The second pseudo-arclength continuation variant is found by also neglecting the terms $V_p^T M V_q$, $c_s^T V_q$ and $c_n^T V_q$ in (2.6). We first solve $\Delta\bar{p}$, ΔT and $\Delta\gamma$ from

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T b_T & V_p^T b_\gamma \\ c_s^T V_p & d_{s,T} & d_{s,\gamma} \\ c_n^T V_p & d_{n,T} & d_{n,\gamma} \end{bmatrix} \begin{bmatrix} \Delta\bar{p} \\ \Delta T \\ \Delta\gamma \end{bmatrix} = - \begin{bmatrix} V_p^T r \\ s \\ n \end{bmatrix}$$

and then compute

$$\Delta q = V_q \sum_{i=0}^{l-1} \left(V_q^T M V_q\right)^i V_q^T \left(r + \Delta\gamma\, b_\gamma\right).$$

This variant is denoted as $CRP(l)$ for continuation variant of the recursive projection method with l Picard iteration steps, since $CRP(1)$ corresponds to the method for continuation in [24].

3.3. Convergence behavior of $NPGS(l)$ and $NPJ(l)$. In [15], we showed that the $NPGS(l)$ and $NPJ(l)$ schemes have linear asymptotic convergence rate $|\mu_{p+1}|^l$ (where μ_{p+1} is the largest eigenvalue smaller than the threshold ρ for the basis V_p). However, this result has several limitations, since a perfect basis V_p is assumed, and it only gives the asymptotic convergence rate, but no information about the transient behavior. Moreover, the proof does not generalize to the $CNP(l)$ and $CRP(l)$ schemes.

We now study the convergence behavior of these methods using (2.15) and (2.16) and we illustrate the discussion by analyzing the convergence observed in actual computations.

Neglecting $V_q^T b_T$ in the $NPGS(l)$ and $NPJ(l)$ methods corresponds to setting $\Delta\bar{q}_T = 0$ in the algebraic framework described in the previous section.

For the $NPGS(l)$ method, (2.15) and (2.16) reduce to

$$(3.5)\quad \begin{cases} Qr(x(0)+\Delta x(0), T+\Delta T) \\ \qquad = Qr_r + \Delta T\, Qb_T + QM\Delta p + \mathrm{O}(\Delta q, \Delta p, \Delta T)^2 \\ \qquad = (QMQ)^l\, r + \Delta T\, Qb_T + QM\Delta p + \mathrm{O}(\Delta q, \Delta p, \Delta T)^2 \\ Pr(x(0)+\Delta x(0), T+\Delta T) = \mathrm{O}(\Delta q, \Delta p, \Delta T)^2 \\ s(x(0)+\Delta x(0), T+\Delta T) = \mathrm{O}(\Delta q, \Delta p, \Delta T)^2. \end{cases}$$

- Let us first assume that the Q-projection of the residual $Qr(x(0), T)$ at the beginning of the $NPGS(l)$ step is large compared to the P-projection $Pr(x(0), T)$. Δq, $V_p^T M\Delta q$ and hence the right-hand side of (3.3) are usually of similar size as $Qr(x(0), T)$. Hence one can expect that Δp and ΔT will be at most of the same order of magnitude as $Qr(x(0), T)$. If the basis V_p is sufficiently accurate, the terms $\Delta T\, Qb_T$ and $QM\Delta p$ will be much smaller than the initial Q-projection of the residual. The terms Qr_r and the higher order terms dominate the Q-projection of the residual at the end of the $NPGS(l)$ step. If the higher order terms are sufficiently small, both projections of the residual decrease if l is sufficiently large.
- On the other hand, if initially the P-projection of the residual is much larger than the Q-projection, ΔT and Δp can be quite large and the terms $\Delta T\, Qb_T$ and $QM\Delta p$ may dominate the Q-projection of the residual and limit its improvement, even if the basis is fairly accurate. $\|Qr\|$ may even grow, but if the higher order terms are not too large, the P-projection and the overall residual decrease. After some steps, we return to the first scenario.

Thus we can conclude that the $NPGS(l)$ scheme tries to keep the size of the P-projection of the residual at or below the size of the Q-projection.

This is not the case for the $NPJ(l)$ scheme. In this case we also neglect

the terms $V_p^T M V_q$ and $c_s^T V_q$, and the following relations hold:

$$
\left\{
\begin{array}{l}
Qr(x(0)+\Delta x(0), T+\Delta T) \\
\qquad = Qr_r + \Delta T\, Qb_T + QM\Delta p + \mathrm{O}(\Delta q, \Delta p, \Delta T)^2 \\
\qquad = (QMQ)^l\, r + \Delta T\, Qb_T + QM\Delta p + \mathrm{O}(\Delta q, \Delta p, \Delta T)^2 \\
Pr(x(0)+\Delta x(0), T+\Delta T) = PM\Delta q + \mathrm{O}(\Delta q, \Delta p, \Delta T)^2 \\
s(x(0)+\Delta x(0), T+\Delta T) = c_s^T \Delta q + \mathrm{O}(\Delta q, \Delta p, \Delta T)^2.
\end{array}
\right. \tag{3.6}
$$

- First assume that the Q-projection of the residual at the beginning of the $NPJ(l)$ step is similar in size or larger than the P-projection. Since the right-hand side of the P-system (3.4) is fairly small, we can expect that Δp and ΔT and hence the terms $\Delta T\, Qb_T$ and $QM\Delta p$ in (3.6) will be small. $\|Qr\|$ can decrease substantially if the higher order terms are small enough and if l is large enough. However, Δq will be large (typically of comparable size as the Q-projection of the initial residual). Hence the term $PM\Delta q$ usually is large (and depending on the phase condition use, $c_s^T \Delta q$ also) and limits the decrease of the P-projection of the residual. In fact, $\|Pr\|$ may even grow.
- On the other hand, if initially the P-projection of the residual is much larger than the Q-projection, Δq and hence also $PM\Delta q$ and $c_s^T \Delta q$ are small. Pr will decrease substantially provided the higher order terms are small. However, since Δp and ΔT are relatively large, the reduction of $\|Qr\|$ is limited by the size of the terms $\Delta T\, Qb_T$ and $QM\Delta p$. In fact, $\|Qr\|$ may even grow.

In general, the $NPJ(l)$ scheme tends to keep $\|Qr\|$ much smaller than $\|Pr\|$, but this is only possible if the basis is accurate enough. Otherwise the scheme continuously switches between both scenarios described above and will alternately reduce $\|Qr\|$ and $\|Pr\|$. Therefore the $NPJ(l)$ needs a more accurate basis than the $NPGS(l)$ method for a similar performance in terms of number of Newton–Picard iterations.

These points are illustrated in Figure 1 and 2. Figure 1 shows the convergence history for the $NPGS(8)$ method and Figure 2 for the $NPJ(8)$ method. Both computations were done for the same (stable) periodic orbit and using approximately the same starting value. The letter B marks the beginning of the Newton–Picard step, the numbers 1 to 8 the 8 Picard steps and E the end of the step. After each Picard step, we updated the point $x(0)$ and computed the residual r to show how the residual and its projections evolve during the iterations.

In these computations, we allowed the dimension p of the subspace $\mathcal{U}$ (or the value of ρ in assumption 2.1) to change between every Newton–Picard step, depending on the number of basis vectors that were considered to be accurate enough (see section 5). The changes of the basis size explain the jumps of the norms of the projections of the residual between successive Newton–Picard steps.

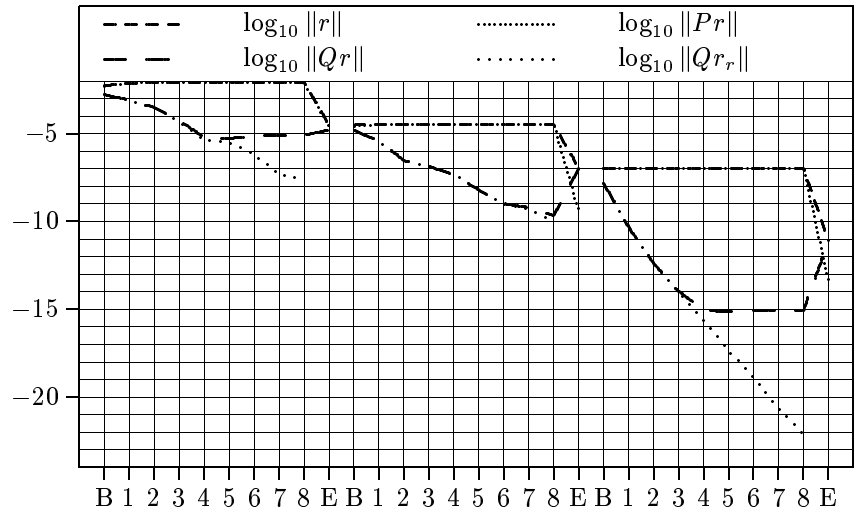

FIG. 1. *Convergence history of the NPGS(8) method. The basis size used for the first step was 2 and for the next steps 3 and 6.*

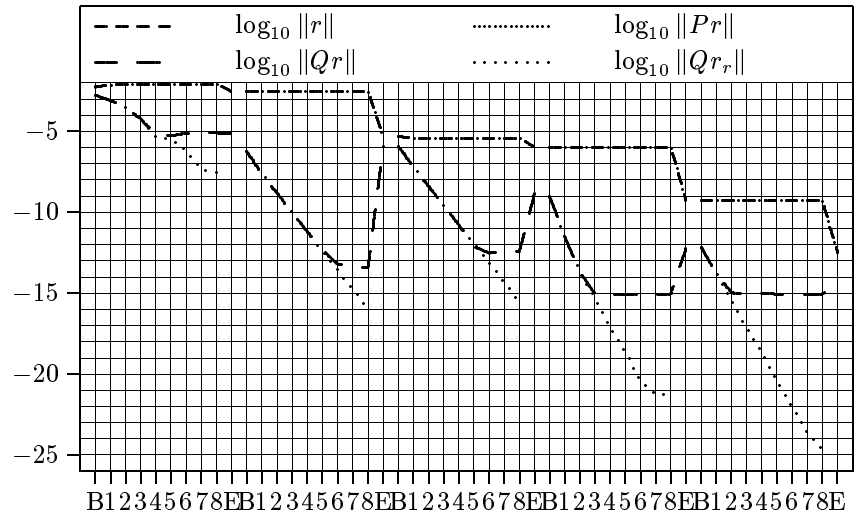

FIG. 2. *Convergence history of the NPJ(8) method. The basis size used for the first step was 2 and for the next steps 5, 5, 6 and 6.*

Notice the approximately linear convergence of the Picard iterations. The convergence factor is different in different Newton–Picard steps since the dimension of $\mathcal{U}$ varies. Initially the curves for $\|Qr\|$ and $\|Qr_r\|$ fall together, but after a while, $\|Qr\|$ does not converge any further. This is explained by the higher order terms in the expansion (2.15). In the fourth Newton–Picard step in Figure 2 the convergence of $\|Qr_r\|$ breaks

off because we have reached the level of roundoff errors. During the initial Picard steps $\|Pr\|$ changes because of the term $PM\Delta q$. As the Picard iterations converge, Δq settles down to a more or less constant value and so does $\|Pr\|$.

During the Newton step (from 8 to E in the figures), $\|Qr\|$ sometimes increases substantially. In the first Newton–Picard step in Figure 1 the higher order terms dominate and no jump is observed. The jump during the second iteration is caused by the term $\Delta T\, Qb_T$ and the jump during the last step by $QM\Delta p$. A similar scenario is observed in Figure 2. Initially, higher order terms dominate the Q-projection in (3.6), then $\Delta T\, Qb_T$ and during the final iterations $QM\Delta p$.

In Figure 1 we see that the $NPGS(8)$ method converged in three steps. In this case, the convergence speed was clearly limited by the accuracy of the basis and by the size of the term Qb_T. Figure 2 for the $NPJ(8)$ method shows a different picture. This scheme needed five steps to converge. We clearly notice the irregular convergence behavior predicted above. During the initial step, the P-projection of the residual does hardly decrease because of the term $PM\Delta q$, but the Q-projection decreases nicely as it did in the $NPGS$ method. In the second step, $\|\Delta q\|$ is much smaller and the P-residual decreases a lot. However, since Δp and ΔT are large, the terms $\Delta T\, Qb_T$ and $QM\Delta p$ are large and prevent the decrease of $\|Qr\|$. $\|Qr\|$ even grows. The third step shows a similar picture as the first step. In the fourth step, the basis is accurate enough and the term Qb_T is small enough so that both projections of the residual can decrease.

In [14] a similar analysis is presented for the $NPGS(1)$ and $NPJ(1)$ methods, where only 1 Picard step is used to solve the Q-systems. In this case, the convergence of the Q-projection of the residual is limited by the convergence of the Picard scheme. In the $NPGS(1)$ method, the P-projection of the residual is much below the Q-projection, while in the $NPJ(1)$ method the convergence of the P-projection is completely governed by the term $PM\Delta q$ and the Q-projection of the residual is kept well below the P-projection.

In general, the $NPGS(l)$ scheme will profit more than the $NPJ(l)$ scheme from a large number of Picard steps l . The latter scheme needs a more accurate basis before it can profit from a larger value of l and from a smaller residual Qr_r since the update Δp and hence $QM\Delta p$ are usually larger than in the $NPGS(l)$ scheme.

3.4. Convergence behavior of $CNP(l)$ and $CRP(l)$. The picture for the continuation variants is not so different. For the $CNP(l)$ scheme

we get

$$(3.7)\quad \begin{cases} Qr(x(0)+\Delta x(0),T+\Delta T,\gamma+\Delta\gamma) = Qr_r + \Delta T\, Qb_T \\ \qquad +\Delta\gamma\, Qr_\gamma + QM\Delta p + \mathrm{O}(\Delta q,\Delta p,\Delta T,\Delta\gamma)^2, \\ Pr(x(0)+\Delta x(0),T+\Delta T,\gamma+\Delta\gamma) = \mathrm{O}(\Delta q,\Delta p,\Delta T,\Delta\gamma)^2, \\ s(x(0)+\Delta x(0),T+\Delta T,\gamma+\Delta\gamma) = \mathrm{O}(\Delta q,\Delta p,\Delta T,\Delta\gamma)^2, \\ n(x(0)+\Delta x(0),T+\Delta T,\gamma+\Delta\gamma) = \mathrm{O}(\Delta q,\Delta p,\Delta T,\Delta\gamma)^2, \end{cases}$$

and for the $CRP(l)$ method

$$(3.8)\quad \begin{cases} Qr(x(0)+\Delta x(0),T+\Delta T,\gamma+\Delta\gamma) = Qr_r + \Delta T\, Qb_T \\ \qquad +\Delta\gamma\, Qr_\gamma + QM\Delta p + \mathrm{O}(\Delta q,\Delta p,\Delta T,\Delta\gamma)^2, \\ Pr(x(0)+\Delta x(0),T+\Delta T,\gamma+\Delta\gamma) = \\ \qquad PM\Delta q + \mathrm{O}(\Delta q,\Delta p,\Delta T,\Delta\gamma)^2, \\ s(x(0)+\Delta x(0),T+\Delta T,\gamma+\Delta\gamma) = \\ \qquad c_s^T\Delta q + \mathrm{O}(\Delta q,\Delta p,\Delta T,\Delta\gamma)^2, \\ n(x(0)+\Delta x(0),T+\Delta T,\gamma+\Delta\gamma) = \\ \qquad c_n^T\Delta q + \mathrm{O}(\Delta q,\Delta p,\Delta T,\Delta\gamma)^2, \end{cases}$$

where

$$Qr_r = (QMQ)^l Qr,$$
$$Qr_\gamma = (QMQ)^l Qb_\gamma.$$

The $CNP(l)$ method will converge keeping $\|Pr\|$ at or below $\|Qr\|$, while the $CRP(l)$ method will keep $\|Qr\|$ below $\|Pr\|$ and converge irregularly if the basis is not accurate enough.

There is however an important difference in the behavior of the fixed parameter and continuation variants if the nonnormality causes initial growth of the residual Qr_γ in (2.13). The analysis presented in [14] shows that in such a case both continuation variants can fail for an arbitrary fixed value of l, even for a starting value arbitrary close to the limit cycle. This indicates that it might be impossible to proof an asymptotic convergence result for the $CNP(l)$ and $CRP(l)$ schemes without additional conditions on the matrix M and/or the vector b_γ. However, this is a rather hypothetical scenario; we have never observed growth of the residuals (2.13) during the initial Picard steps. But we learn from the analysis in [14] that in a robust code l should be allowed to vary and the size of the term Qr_γ should be monitored.

4. Developing a robust method.

In this section, we first recapitulate the different terms in (2.15) and (2.16) that limit the convergence. Next we discuss how convergence problems caused by each of those terms can be treated. We then show that the relations (2.15) and (2.16) can be exploited in two complementary ways. They can be used at the beginning of a Newton–Picard step ("a priori")

to determine convergence thresholds for the basis and the various iterative solvers for the Q-systems and they can be used at the end of the iteration step ("a posteriori") to analyze the convergence behavior. Combining both strategies leads to powerful, efficient and robust methods.

4.1. Factors limiting convergence. From (2.15) and (2.16) it is clear that five factors limit the improvement of the residual that can be obtained from a given Newton–Picard step.

1. Higher-order terms neglected by the Newton linearization put limits on both the Q- and the P-projection of the final residual.
2. Inaccuracy of the basis (the term $QM\Delta p$) limits the Q-projection of the final residual.
3. The accuracy of the solutions of the Q-systems (2.7) or (2.10) influences the Q-projection of the final residual.
4. If the P-system (2.8) or (2.11) is solved inaccurately, the term (2.17) shows up in (2.16) and limits the convergence of the residuals Pr, s and n of the nonlinear iteration.
5. Numerical errors, especially in the evaluation of the matrix–vector products and derivatives, but also in other operations, can also limit the convergence.

4.2. Curing convergence problems. Robust methods will must control each of these elements.

1. Higher-order terms can be controlled using damping strategies (see, e.g., [1]). Note that this is not very important if the solver is used in a continuation code. If the higher order terms are too large, the solver fails and the predictor decreases the steplength and generates a new and better starting value. However, damping is useful to compute a single solution or to start the continuation run if good starting values are not available.
2. Influence of the basis inaccuracy. An inaccurate basis can limit the improvement of the Q-projection of the residual. We must make sure that all components of the basis are accurate enough. One inaccurate basis vector can significantly hurt the performance of the Newton–Picard scheme.
3. The accuracy of the solution of the Q-systems. To avoid problems with initial growth of the residuals (2.13), one should use a variable number of Picard steps and test for convergence after each iteration step. The convergence test is very cheap, since the result of the matrix–vector product that is needed for the test can be used in the next Picard step or to build the P-system. One can also consider to solve the Q-systems using more advanced iterative methods. We will also discuss the choice of good starting values for the iterative methods.
4. The accuracy of the solution of the P-system. This is a low-dimensional system and the best direct method should be used

to solve this system, e.g., a least-squares based procedure.

5. Numerical errors can only be controlled by using high-quality time integration techniques and robust, stable numerical methods at each step in the algorithm.

4.3. A posteriori use of the convergence relations. At the end of a Newton–Picard iteration step, the new residual is computed. If the convergence goal for the iteration step is not reached, (2.15) and (2.16) can be used to analyze what went wrong and to take the appropriate action.

- If the higher order terms caused divergence, a damping strategy should be used or the Newton–Picard solver should return with an error condition and rely on the continuation code to generate a better starting value (by decreasing the stepsize).
- If the size of $QM\Delta p$ causes problems, the quality of the basis should be improved and the solution of the Q-systems should be recomputed. Refining the basis is expensive and should only be done if the Newton–Picard step diverged or if a sufficient additional decrease of the residual is possible.
- If one of the residuals (2.13) caused problems, some more Picard iterations to solve the Q-systems should be done.

Remark that we can already test the size of the residuals (2.13) and $QM\Delta p$ in (2.15) before the computation of the new residual $r(x(0)+\Delta q+\Delta p, T+\Delta T, \gamma+\Delta\gamma)$ to check whether these terms have decreased enough. The early detection of problems can avoid a superfluous integration of (1.1). This is particularly important if the integration of (1.1) is expensive compared to the matrix–vector products.

4.4. A priori use of the convergence relations. At the beginning of a Newton–Picard iteration step, we can use (2.15) and (2.16) to determine the convergence thresholds for the computation of the basis V_p (see section 5) and for the iterative solvers for (2.7) and (2.10), provided one has good estimates for $\|\Delta p\|$, $|\Delta T|$ and $|\Delta\gamma|$. Based on the initial residual and estimates for the higher order terms, the goal for the residual at the end of the Newton–Picard step is determined. Based on the estimates for $\|\Delta p\|$, $|\Delta T|$ and $|\Delta\gamma|$ and the goal for the step, it is possible to derive from (2.15) convergence thresholds for the iterative solver and computation the basis V_p. We now discuss two elements from this strategy in more detail. First, we discuss the estimation of $\|\Delta p\|$, $|\Delta T|$ and $|\Delta\gamma|$. Next we propose a strategy to determine the goal for the residual at the end of a Newton–Picard step.

4.5. Estimating the size of Δp, ΔT and $\Delta\gamma$. We can estimate $\|\Delta p\|$, $|\Delta T|$ and $|\Delta\gamma|$ based on the vectors b_T and b_γ and the residual r. A very rough estimate can be obtained from (2.2). $\|\Delta p\|$ and the total update $\|\Delta x(0)\|$ are usually of the same order of magnitude, even if the Q-projection of the residual at the beginning of the step is much larger

than its P-projection. Because of the term $V_p^T M \Delta q_r$ in the right-hand side of the P-system (2.8), the initial residual of the Newton–Picard step and the right-hand side of the P-system are usually of the same size. We estimate that $\|\Delta p\|$ is of the same order of magnitude as the norm of the initial residual, i.e.,

$$\|\Delta p\| \approx \|r\| . \tag{4.1}$$

This estimate may be too pessimistic (i.e., overestimate the norm of $\Delta x(0)$) if the limit cycle is very unstable. In the latter case, a small error in $x(0)$ can cause a large residual. The estimates for $|\Delta T|$ and $|\Delta \gamma|$ are derived from comparing the right-hand side of (2.2) with the columns corresponding to ΔT and $\Delta \gamma$, e.g.,

$$|\Delta T| \approx \frac{\|r\|}{\|b_T\|}, \quad |\Delta \gamma| \approx \frac{\|r\|}{\|b_\gamma\|}. \tag{4.2}$$

We stress that these estimates are very crude and more work needs to be done to develop better estimates.

4.6. Setting the convergence goal. Setting the convergence goal for the Newton–Picard step is even harder. We determine a factor ρ_{goal} expressing the desired improvement of the residual, i.e., at the end of the Newton–Picard step we wish to obtain

$$\|r(x(0) + \Delta x(0), T + \Delta T, \gamma + \Delta \gamma)\| \approx \rho_{goal} \|r(x(0), T, \gamma)\| . \tag{4.3}$$

We will always ensure $\rho_{goal} \leq \rho_{req}$, a user-determined parameter. To determine ρ_{goal}, we proceed as follows. First, we estimate the higher order terms in (2.15) and (2.16) based on the results of the previous Newton–Picard steps. If the estimated higher order terms are larger than $\rho_{req} \|r(x(0), T, \gamma)\|$, we set $\rho_{goal} = \rho_{req}$. It is possible that the criterion (4.3) cannot be met in this case, even if a damping strategy would be used. If the estimated higher order terms are smaller than $\rho_{req} \|r(x(0), T, \gamma)\|$ we have more options. By changing the strategy to determine ρ_{goal}, the Newton–Picard iterations can show either linear or quadratic convergence behavior (or any behavior in between). Quadratic convergence can be obtained by reducing $\|r\|$ in each iteration step to the level of the estimated higher order terms. This requires a lot of work in each step, but reduces the number of Newton–Picard iterations. This strategy makes sense if the matrix–vector products are much cheaper than the integration of the nonlinear system (1.1). We may also settle for linear convergence of the Newton–Picard iterations, or something in between linear and quadratic convergence, say

$$\rho_{goal} = \rho_{req}^{1-\omega} \xi^{\omega}, \quad 0 \leq \omega \leq 1,$$

where ξ is the ratio of the estimated higher order terms to the initial residual. $\omega = 0$ corresponds to linear convergence with convergence rate ρ_{req}

and $\omega = 1$ corresponds to quadratic convergence. Once we have determined ρ_{goal}, we set the goal for the Q-projection of the final residual and for the individual residuals (2.13) and the basis iterations. We must also put a lower limit on ρ_{goal} since the obtainable improvement is limited by the errors in the computation of the matrix–vector products and not only by the higher order terms, and a wrong estimate for the higher order terms may lead to an infeasible goal. Hence we limit

$$\rho_{best} \leq \rho_{goal} \leq \rho_{req}$$

where ρ_{best} is a user-determined parameter. Furthermore, to avoid unnecessary work, the residual should not decrease much below the convergence requirement for the Newton–Picard procedure. This determines another lower limit for ρ_{goal}.

4.7. Minimizing the total cost. The parameter ρ in Assumption 2.1 is another important parameter to be determined. Lower values of ρ result in more basis vectors and thus more work to construct the basis, but a faster convergence of the Picard iterations used to compute the vectors $\Delta\bar{q}_*$. The ideal strategy for the choice of ρ and ρ_{goal} should minimize the total cost for the computation of the periodic orbit. However, it is impossible to predict the total cost of the iterations in advance in terms of the various parameters. A less ambitious goal is to minimize the ratio of the computational cost to the improvement of the residual at each step. This is also impossible since one cannot determine in advance the precise cost of the subspace and Picard iterations, since this depends on the full (unknown) spectrum of the matrix M.

5. Computing the basis.

A crucial aspect of the Newton–Picard method is the efficient calculation and repeated updating of the basis V_p for the low-dimensional subspace $\mathcal{U}$. Recall that $\mathcal{U}$ is the space spanned by the (generalized) eigenvectors of M corresponding to eigenvalues greater than ρ in modulus. Since we are only interested in the dominant eigenvalues and since we are able to compute matrix–vector products with the matrix M, subspace iteration can be used. This is also proposed in [9, 10, 11] and [24]. In order to keep the number of (expensive) matrix–vector multiplications low, we use the most sophisticated version of this algorithm, namely, subspace iteration with projection after each iteration and locking (deflation).

5.1. Subspace iteration with projection and locking. Subspace iteration computes an orthonormal basis for the eigenspace $\mathcal{U}$, corresponding to the p most dominant eigenvalues of a matrix M. Let

$$V^{[0]} = \left[\begin{array}{ccc} v_1^{[0]} & \cdots & v_p^{[0]} \end{array} \right],$$

where $\left\{v_1^{[0]}, \ldots, v_p^{[0]}\right\}$ is an orthonormal basis for an initial guess for the subspace $\mathcal{U}$. The convergence of subspace iteration can be accelerated by using p_e additional vectors, starting the algorithm with the matrix

$$V_e^{[0]} = \left[\begin{array}{cccccc} v_1^{[0]} & \cdots & v_p^{[0]} & \cdots & v_{p+p_e}^{[0]} \end{array} \right],$$

where $\left\{v_1^{[0]}, \ldots, v_{p+p_e}^{[0]}\right\}$ is an orthonormal basis for an initial guess for the dominant invariant subspace of dimension $p + p_e$. We use the following algorithm, which is a variant of Algorithm 5.4 in [22]. Because of the projection step, the basis vectors corresponding to larger eigenvalues converge faster. Locking is used to avoid the updating of sufficiently converged basis vectors [12].

ALGORITHM 1. *Subspace iteration with projection and locking.*

Input:

$V_e^{[0]} = \left[\begin{array}{cccccc} v_1^{[0]} & \cdots & v_p^{[0]} & \cdots & v_{p+p_e}^{[0]} \end{array} \right]$

routine to compute Mv.

Output:

orthonormal matrix V_e, *where* $\mathrm{Span}(V_e[1{:}p])$ *is a good approximation for* $\mathcal{U}$;

$W = MV_e$ *and* $S = V_e^T W$.

begin

$p_{eff} \leftarrow 0$

$W = V_e^{[0]}$

repeat

$V_e[p_{eff}+1{:}p+p_e] \leftarrow W[p_{eff}+1{:}p+p_e]$

Orthonormalize the column vectors of V_e *starting at column* $p_{eff}+1$.

Compute $W[p_{eff}+1{:}p+p_e] = MV_e[p_{eff}+1{:}p+p_e]$.

Compute $U = V_e^T MV_e = V_e^T W$

Compute the Schur decomposition $UY = YS$ *of* U, *order the Schur vectors according to decreasing modulus of the corresponding eigenvalue.*

$V_e \leftarrow V_e Y$, $W \leftarrow WY$.

Determine the number p_{eff} *of accurate vectors in* V_e.

until $p_{eff} = p$

end

In [22] locked vectors are not updated: U is computed as $U = V_e[p_{eff}+1{:}p_e]^T W[p_{eff}+1{:}p+p_e]$ and the step $V_e \leftarrow V_e Y$, $W \leftarrow WY$ becomes $V_e[p_{eff}+1{:}p] \leftarrow V_e[p_{eff}+1{:}p]Y$, $W[p_{eff}+1{:}p] \leftarrow W[p_{eff}+1{:}p]Y$. We have chosen another approach and allow small updates to already locked vectors based on information gained from the new matrix–vector products of the other basis vectors. This is similar to the procedure used in LOPSI [12]. This approach allows reordering of the eigenvalues in the Schur decomposition and takes into account the (unlikely) case that an eigenvalue

is initially underestimated, but converges to a value that is larger than a locked eigenvalue.

We use $p_e \geq 2$. This ensures that μ_{p+1} converges in non-degenerate cases, since μ_{p+1} can be a complex eigenvalue. Convergence of the eigenvalue μ_{p+1} is important in our strategy to select the value of p (see below). Since the convergence factor of a simple eigenvalue μ_i is $|\mu_{p+p_e}| / |\mu_i|$, a larger value of p_e greatly improves the convergence speed of the p dominant eigenvalues. This compensates for the additional cost per subspace iteration step due to the use of a larger basis.

In each Newton-Picard iteration step, the basis V_p must be (re)-computed. If l is small and if ρ is rather large, we do not need a very accurate basis; and if a good starting value for V_p is available, one or two subspace iteration steps per Newton–Picard step are usually sufficient. Otherwise, e.g., before the first Newton–Picard step, more subspace iterations are needed. Locking is then very useful if some of the Floquet multipliers are large, since the corresponding basis vectors will converge to the required accuracy in few subspace iteration steps, while for the other vectors, more iteration steps are needed. In our code converged eigenvalues remain locked within a Newton–Picard step, but locking is normally not maintained over successive Newton–Picard steps. Indeed, preliminary locking of vectors and failing to unlock them again can lead to slow convergence or divergence of the algorithm. Furthermore, there are theoretical objections against this procedure, since M changes between two Newton–Picard iteration steps and W is no longer exactly equal to MV for the locked columns. In any case, all vectors are unlocked to compute accurate values for the Floquet multipliers after the final Newton–Picard step.

We now discuss the convergence criterion we used, the choice of the initial basis and a strategy for the determination of the basis size.

5.2. Convergence criterion. Our aim is to compute a (nearly) invariant subspace Span(V_p) of M and to ensure that the term $V_q^T M V_p \Delta\bar{p}$ which is neglected in (2.6) remains small. As can be seen from (2.15) and Figure 1, it is important to control the size of the latter term. The convergence criterion for the subspace iteration is based on [25] and requires no new matrix–vector products with M. Let $V_k = V_e[1{:}k]$, $W_k = W[1{:}k]$ and $S_k = S[1{:}k, 1{:}k]$. The number p_{eff} of vectors that are locked in Algorithm 1 is determined to be the largest value of k for which $S[k+1, k] = 0$ (i.e., the (k, k) element of S is not the upper left element of a 2×2-block) and

$$\zeta_k = \max_{\substack{\|\bar{p}_k\|_2 = 1 \\ \bar{p}_k \in \mathbb{R}^k}} \left\| \left(I - V_k V_k^T\right) M V_k \bar{p}_k \right\|_2$$

is lower than a user-determined threshold ε_{sub} for the basis accuracy. ζ_k is the largest singular value $\sigma_{\max}$ of

$$Z_k = (I - V_k V_k^T) M V_k = W_k - V_k V_k^T W_k = W_k - V_k S_k. \tag{5.1}$$

Note that Z_k can be computed without new matrix–vector products with M and that (5.1) is only valid if $S[k+1,k] = 0$. Under the latter condition, we also have

$$Z_k = Z_p[1{:}k]. \tag{5.2}$$

Z_k is a $N \times k$-matrix and we wish to avoid doing many computations with it. Because of implementation issues, we only allow AXPY-operations and scalar products with N-dimensional vectors. Furthermore, we wish to determine $\zeta_k = \sigma_{\max}(Z_k) = \sqrt{\lambda_{\max}(Z_k^T Z_k)}$ for different values of k. Note that

$$Z_k^T Z_k = \left(W_k^T - S_k^T V_k^T\right)(W_k - V_k S_k) = W_k^T W_k - S_k^T S_k$$

If we first compute $A = W^T W$, then

$$Z_k^T Z_k = A[1{:}k, 1{:}k] - S_k^T S_k,$$

and we can compute $\zeta_k = \sigma_{\max}(Z_k)$ efficiently for different values of k. Although computing the matrix $A^T A$ explicitly and then computing its eigenvalues is not a very stable algorithm, this approach is accurate enough for our purpose and it does not require the construction of the matrix Z_k (for an alternative approach, see [14]).

5.3. The initial basis. If a branch of periodic solutions is computed, the final basis for the periodic solution at the previous continuation point can be used as the initial guess for the basis $V_e^{[0]}$. To improve the robustness of the calculation of the Floquet multipliers, components of $V_e^{[0]}$ are randomly perturbed at the start of each new continuation step. This helps to ensure that new eigenvectors can enter the basis rapidly and that all eigenvalues outside of the disk C_ρ are computed.

For the first point on a branch we can either perform some subspace iterations starting with random vectors or derive starting values from the bifurcation point where the branch originates from. For example, at a Hopf point with critical eigenvalues $\pm i\omega$, the Floquet multipliers of the originating periodic orbit are given by

$$\mu_i = e^{\frac{2\pi\lambda_i}{\omega}}$$

where λ_i are the eigenvalues of f_x at the Hopf point. The initial basis is then given by the Schur vectors corresponding to the $p + p_e$ rightmost eigenvalues of f_x. These eigenvalues and the corresponding basis can be computed using appropriate iterative techniques, see, e.g., [17].

5.4. Basis size. During the subspace iteration, the dimension p of the subspace $\mathcal{U}$ must be updated such that

$$|\mu_1| \geq \cdots \geq |\mu_p| > \rho > |\mu_{p+1}| \geq \cdots \geq |\mu_{p+p_e}|.$$

Choosing the value of p too large does not cause trouble. However, if p is too small and if $|\mu_{p+1}| \geq 1$, the Picard iterations used to solve the Q-systems (2.7) will fail to converge. To ensure early detection of growth of the size of the basis, we use a weaker threshold in the convergence criterion for the decision on adding and removing basis vectors than in the criterion for the locking of vectors. Suppose $\mu_1, \ldots, \mu_i$ would be locked if the weaker threshold were used and suppose n_{vec} is the current number of vectors in the basis. Let p_{large} and be the number of eigenvalues in the set $\{\mu_1, \ldots, \mu_i\}$ that are larger than ρ. Let p_{small} be the number of eigenvalues that are smaller than $r_{hist}\rho$ with $r_{hist} < 1$. ($0.8 < r_{hist} < 0.9$ adds some hysteresis to the criterion to avoid that we quickly add again a vector that has been removed or vice versa.) Vectors must be added to the basis if $p_{large} + p_e > n_{vec}$. Vectors should be removed if $p_{small} > 0$ and $(k-1) + p_e < n_{vec}$, with k such that μ_k is the largest eigenvalue in the set $\{\mu_1, \ldots, \mu_i\}$ that is smaller than $r_{hist}\rho$. Note that some test results in section 8 are computed with a version of the code in which vectors were only removed if $p_{small} > p_e$ (a too conservative criterion).

The combination of the use of extra vectors, random perturbations of the starting values and our criterion for adding and removing vectors proved to be reliable for our test cases. It is much simpler and more reliable than criteria based on monitoring the convergence of the Newton-Picard iteration, as used in [9, 24]. Indeed, slow convergence can also be caused by an inaccurate basis or bad starting values. Increasing the value of p does not the appropriate action in those cases and leads to a too large value of p. The extra cost of using p_e extra vectors is largely compensated by the higher convergence speed of the subspace iteration.

6. Solving the Q-systems.

From the discussion in the previous sections, we can deduce the following requirements for the Q-system solver:

1. The Q-system solver must be able to use a variable number of steps, controlled by a convergence criterion.
2. The solver must be able to start with a nonzero starting value, since it is possible to derive good starting values for $\Delta\bar{q}_r$ in (2.7) — see below.
3. The solver should also output the terms $M\Delta q_*$ or $V_p^T M\Delta q_*$ for the construction of the P-system. These terms can be recovered easily from the Q-system solver. We save a matrix–vector product for the construction of the P-system in doing so.
4. During the a posteriori analysis we may decide that the solution of one of the Q-systems needs further refinement. Therefore the Q-system solver should generate the necessary information to make the restart as cheap as possible.

We will first discuss the use of a Picard iteration scheme to solve the Q-systems (2.7) or (2.10). Then we will discuss the choice of the starting values for the various Q-systems.

6.1. The Picard scheme. In [15, 20], we proposed to compute Δq using a fixed number of Picard iterations. With a variable number of steps, the Picard iteration is given by

$$\begin{cases} \Delta\bar{q}_* \leftarrow \Delta\bar{q}_*^{[0]}, \\ \Delta\bar{q}_* \leftarrow V_q^T M V_q \Delta\bar{q}_* + V_q^T r_* \textbf{ until} \text{ convergence}, \end{cases}$$

where r_* denotes the residual r, b_T or b_γ and where $\Delta\bar{q}_*$ denotes the corresponding unknown in (2.7) or (2.10). The use of the matrix V_q can be avoided by premultiplying each term with V_q and instead compute the N-dimensional vectors $\Delta q_* = V_q \Delta\bar{q}_*$, i.e.,

$$\begin{cases} \Delta q \leftarrow \Delta q^{[0]}, \\ \Delta q \leftarrow Q M \Delta q + Q r_* \textbf{ until} \text{ convergence}. \end{cases} \tag{6.1}$$

The scheme has converged if

$$\|Q\,(r_* + (M - I)\Delta q_*)\| \le \varepsilon,$$

where ε is the convergence threshold. This test is basically free since the matrix–vector product $M\Delta q_*$ can be used in the next Picard step or to construct the P-system. Our algorithm must take into account two different start scenarios:

1. A starting vector Δq_* is given, but $M\Delta q_*$ is not yet known.
2. Both Δq_* and $M\Delta q_*$ are given. This occurs if the starting value is 0 since then $M\Delta q = 0$ and at a restart.

This leads to the following algorithm. Note that we have omitted the subscript $*$ from our notation in the algorithm.

ALGORITHM 2. *Picard iteration for single shooting.*

Input

Starting value $\Delta q^{[0]}$.
Optional: $M\Delta q^{[0]}$.
Convergence threshold ε.
Routine to compute Mv.
Basis V_p.
Right-hand side r.

Output

$\Delta q = V_q \Delta\bar{q}$ *with* $\Delta\bar{q}$ *an approximate solution to* $\left(V_q^T M V_q - I_q\right)\Delta\bar{q} = -V_q^T r$.
$M\Delta q$.

begin

if $\Delta q^{[0]} = 0$ ***then***
$\Delta q \leftarrow 0$; $M\Delta q \leftarrow 0$;

else if $M\Delta q^{[0]}$ *is known*
$\Delta q \leftarrow \Delta q^{[0]}$; *Set* $M\Delta q = M\Delta q^{[0]}$.
else
$\Delta q \leftarrow \Delta q^{[0]}$;
Compute $M\Delta q$.
endif
$RES \leftarrow \left(I - V_p V_p^T\right)(r + M\Delta q - \Delta q)$
while $\|RES\| > \varepsilon$ ***do***
$\Delta q \leftarrow \left(I - V_p V_p^T\right)(M\Delta q + r)$
Compute $M\Delta q$.
$RES \leftarrow \left(I - V_p V_p^T\right)(r + M\Delta q - \Delta q)$
end while
end

Note that this algorithm may return without actually updating Δq if the initial solution is accurate enough. This is important if Δq_T is being computed since the term Qb_T is often small enough so $\Delta q_T = 0$ is sufficient. In the actual implementation, we also stop if a predetermined maximum number of iterations is exceeded. l steps of this algorithm require l matrix–vector products if $M\Delta q^{[0]}$ is known and $l+1$ products otherwise.

6.2. Starting values. Starting values are needed at the beginning of the Newton–Picard step and at a restart after the a posteriori analysis. Let us first consider the latter case.

6.2.1. Starting values for a restart. If the basis V_p has not changed, we can restart Algorithm 2 using the values of Δq_* and $M\Delta q_*$ returned after the previous run for the same right-hand side. The situation is different if the basis has been refined. We need to ensure that Δq_* lies in the new subspace $\mathcal{U}^\perp$. Therefore we will reproject Δq_* and at the same time make the corresponding updates to $M\Delta q_*$:

$$\begin{cases} \Delta q_* \leftarrow \left(I - V_p V_p^T\right)\Delta q_*, \\ M\Delta q_* \leftarrow M\Delta q_* - (MV_p)V_p^T\Delta q_*. \end{cases} \tag{6.2}$$

Note that MV_p is known from the subspace iterations. Formula (6.2) produces a good starting value if vectors that were contained in V_p have not changed much and if the basis size has not decreased (i.e., no new vectors added to V_q). To cope with a decrease of the dimension of $\mathcal{U}$ we suggest to add to the vectors Δq_* components in the direction of the eigenvectors corresponding to the smallest of the p dominant eigenvalues before the restart of the subspace iteration and to remove the unnecessary components afterwards using (6.2).

6.2.2. Starting values at the beginning of a Newton–Picard step. At the limit cycle, $r = 0$ and thus $\Delta q_r = 0$. Therefore we use $\Delta q_r^{[0]} = 0$ as starting value for the Q-system with right-hand side Qr.

Although the vector b_T does not change much during the iterations, the right-hand side Qb_T can change considerably since b_T^* is an eigenvector of M^* for the eigenvalue 1 (at least if we neglect the effect of time discretization errors). At the limit cycle, $Qb_T^* = 0$ for a perfect basis and hence $\Delta q_T = 0$. Thus $\Delta q_T^{[0]} = 0$ is a reasonable starting value. Using the result of the previous iteration as a starting value is not a good idea.

Except when the basis size changes, Qb_γ is fairly constant. Hence one can use the value of Δq_γ computed in the previous iteration as starting value and reproject Δq_γ into $\mathcal{U}^\perp$ using (6.2). To cope with the effect of a decrease of the basis size, one can use the same procedure as for the restart after a basis change. At the end of the Newton–Picard step, additional components are added to Δq_γ using the procedure outlined above. The projection (6.2) at the beginning of the next Newton–Picard step again removes unnecessary components.

It is sometimes possible to derive better starting values at the beginning of a Newton–Picard step by using components of the basis V_e in Algorithm 1 that are not used in the basis V_p. This procedure is discussed in [14].

7. Solving the P-system.

The cost of solving the small P-systems (2.8) or (2.11) is only a negligible fraction of the total cost of a Newton–Picard step. If an exact basis V_p and exact solutions of the Q-systems were used, the P-system (2.8) would always be singular at a transcritical or pitchfork bifurcation point (and (2.11) even at a fold point). It is very ill-conditioned in the neighborhood of these points. To improve the numerical stability, the P-system can be solved using the least-squares method [5] instead of Gaussian elimination, since the cost is not a problem.

As an additional advantage of using the least-squares method, the explicit parameterizing equation and phase condition can be omitted. Indeed, the least-squares approach automatically selects good conditions, as we now explain.

Suppose $A \in \mathbb{R}^{m\times n}$ with $n > m$ and $\mathrm{Rank}(A) = m$. The singular value decomposition (SVD) of A is given by

$$A = U\Sigma V^T \tag{7.1}$$

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal matrices, $\Sigma \in \mathbb{R}^{m\times n}$, with $\Sigma[i,i] = \sigma_i$, $\Sigma[i,j] = 0$ if $i \neq j$, and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m > 0$. Note that

$$\mathrm{Ker}(A) \quad = \quad \mathrm{Span}\,(V[m+1:n]) \tag{7.2}$$

The least squares solution of $Ax = b$ is given by

$$x_{LS} = V\Sigma^+ U^T b.$$

with $\Sigma^+ \in \mathbb{R}^{n\times m}$ a matrix with all elements equal to zero except the elements on the main diagonal which are $\sigma_1^{-1}, \ldots, \sigma_m^{-1}$. x_{LS} satisfies

$$\begin{aligned} U^T(b - Ax_{LS}) &= 0 \\ V[m+1:n]^T x_{LS} &= 0. \end{aligned} \tag{7.3}$$

Hence x_{LS} also solves the system

$$\begin{bmatrix} A \\ V[m+1:n]^T \end{bmatrix} x = \begin{bmatrix} b \\ 0 \end{bmatrix}.$$

Since A has full rank, this system is nonsingular and has a unique solution. Let us first assume we only omit the parameterizing equation. This corresponds to $n = m+1$ and the least-squares solution of $Ax = b$ is precisely the solution of

$$\begin{bmatrix} A \\ v_{m+1}^T \end{bmatrix} x = \begin{bmatrix} b \\ 0 \end{bmatrix}.$$

The least–squares approach chooses a parameterization equation which minimizes the vector x. This resembles closely the approach used in CONTENT [13] for the continuation of steady-state solutions of $f(x,\gamma) = 0$.

When both the phase condition and the parameterizing equation are omitted, the least-squares method will automatically choose a rather good phase condition. Let

$$A = \begin{bmatrix} A_0 & b_0 & b_1 \end{bmatrix}, \quad A_0 \in \mathbb{R}^{m\times m}, \quad A_0 b_0 = 0,$$

and suppose 0 is a simple eigenvalue of A_0. The least-squares solution x_{LS} of $Ax = b$ satisfies

$$V[m+1:m+2]^T x_{LS} = 0.$$

Since $\mathrm{Ker}(A) = \mathrm{Span}\{v_{m+1}, v_{m+2}\}$ and since

$$A \begin{bmatrix} b_0 \\ 0 \\ 0 \end{bmatrix} = 0,$$

we also have

$$\begin{bmatrix} b_0^T & 0 & 0 \end{bmatrix} x_{LS} = 0. \tag{7.4}$$

If A_0 is not exactly singular and if b_0 is close to the eigenvector corresponding to the smallest eigenvalue of A_0, then (7.4) is not exactly satisfied, but still $\begin{bmatrix} b_0^T & 0 & 0 \end{bmatrix} x_{LS} \approx 0$. The P-system (2.8) has precisely this property. Suppose V_p is an exact basis, then at the limit cycle

$$V_p^T(b_T + M\,\Delta q_T) = V_p^T b_T$$

is the eigenvector of $V_p^T M V_p - I_p$ for the trivial Floquet multiplier 1. The SVD based least-squares approach implicitly adds a condition which is almost equivalent to the phase conditions

$$b_T^T V_p \Delta\bar{p} = 0 \tag{7.5}$$

or

$$b_T^T \Delta x(0) = 0 \tag{7.6}$$

(since $b_T^T \Delta x(0) = b_T^T V_p \Delta\bar{p} + b_T^T V_q \Delta\bar{q} \approx b_T^T V_p \Delta\bar{p}$). The least-squares approach automatically selects a close to optimal condition, although conditions like (7.5) or (7.6) with zero right-hand side cannot eliminate the phase shift introduced by the predictor in the continuation code. If this is a problem, an explicit phase condition should be used.

8. Test results.

In this section we will discuss some test results for two different problems: the Brusselator model and a model studied by Elezgaray and Arneodo in [3].

8.1. The Brusselator model. The one-dimensional Brusselator reaction–diffusion system is modeled by the equations [8]

$$\begin{cases} \dfrac{\partial X}{\partial t} &= \dfrac{D_X}{L^2}\dfrac{\partial^2 X}{\partial z^2} + X^2 Y - (B+1)X + A, \\ \dfrac{\partial Y}{\partial t} &= \dfrac{D_Y}{L^2}\dfrac{\partial^2 Y}{\partial z^2} - X^2 Y + B \end{cases} \tag{8.1}$$

with Dirichlet boundary conditions

$$\begin{cases} X(t, z=0) = X(t, z=1) = A, \\ Y(t, z=0) = Y(t, z=1) = \dfrac{B}{A}. \end{cases} \tag{8.2}$$

We use the characteristic length L as the bifurcation parameter while the other parameters are fixed at $A = 2$, $B = 5.45$, $D_X = 0.008$ and $D_Y = 0.004$. For these parameter values, branches of periodic solutions bifurcate from the trivial steady state branch ($X = A$, $Y = B/A$) at Hopf bifurcation points at

$$L_k^H = k\pi\sqrt{\frac{D_X + D_Y}{B - A^2 - 1}} = k\, 0.5130$$

[7]. For the reported results, we use an $O(h^2)$ finite difference space discretization with grid size $h = 1/32$, yielding a system of ODEs of dimension $N = 62$. The time integrations were done using the LSODE package [18].

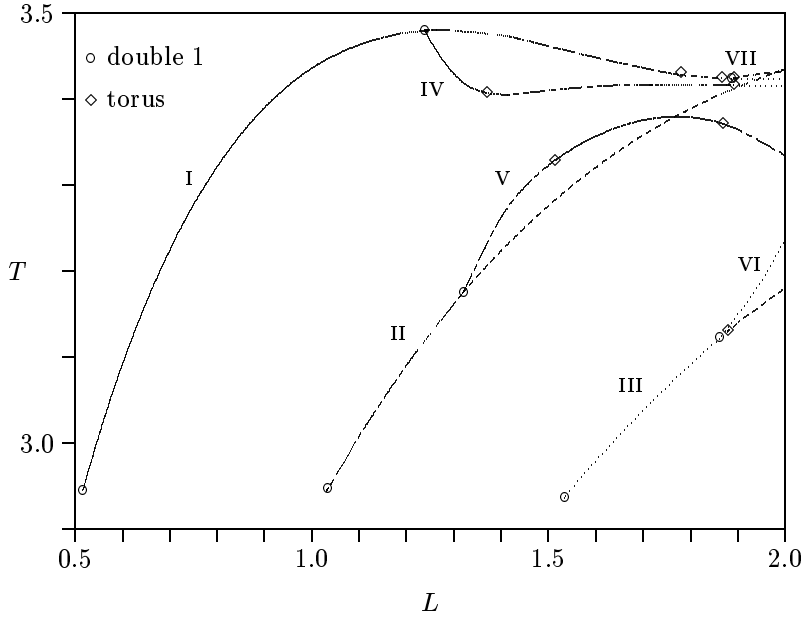

FIG. 3. *Periodic solutions bifurcation diagram for the discretized Brusselator model* ($h = \frac{1}{32}$), *period* T *versus the reactor length* L. *Roman numbers indicate the numbering of the branches used in this section. Double one Floquet multipliers (pitchfork bifurcations and Hopf points) and torus bifurcations are marked with* ○ *and* ◇ *respectively. No period doublings occur on the computed branches.*

The bifurcation diagram for the periodic solutions for L between 0.5 and 2 is shown in Figure 3. Several torus- and pitchfork bifurcation points were detected on the branches. They were located accurately by taking very small stepsizes along the branch. The new branches emanating from the various pitchfork bifurcations were also computed. Note that on branch I the torus bifurcation at $L \approx 1.867$ is immediately followed by a pitchfork bifurcation at $L \approx 1.887$. On the emanating branch VII there is almost immediately a torus bifurcation at $L \approx 1.8904$.

We have compared the efficiency of two Newton–Picard variants, i.e., $NPGS$ (with continuation in the parameter L) and CNP with a full Newton shooting approach to solve (1.3). In the latter approach, the Jacobian matrix is kept fixed during each Newton iteration (Chord-Newton method).

In [15] we presented an extensive set of test results. These test results showed the efficiency of the Newton–Picard approach, but did not show results on the robustness. Each of the results reported was the best result over twelve runs with different choices for the parameters in the algorithm, and many of those runs failed. We will report results for the same four branches as in [15], i.e., branches I, II, III and IV in Figure 3. As in [15], we count each matrix–vector product for one initial value problem (IVP)

TABLE 1
Number of IVP solves for the NPGS and CNP methods, 1-dimensional Brusselator model. For comparison purposes, we also list the result for the Chord–Newton method from [15].

Method	branch I	branch II	branch III	branch IV	total
$NPGS$	996	1102	831	1304	4233
CNP	1194	1132	1051	1361	4738
Chord–Newton	2053	1607	1128	1874	6662

solve.

We have made the following choices in the $NPGS$ and CNP methods.

- In the subspace iteration (Algorithm 1) we used $p_e = 2$ and the threshold to add vectors to the basis was 0.55, the threshold to delete vectors 0.45. We switched to a Newton–Picard step as soon as the Schur basis corresponding to all eigenvalues larger than 0.8 satisfied $\sigma_{max}(Z_k) \leq 10^{-2}$ with Z_k given by (5.1).
- Only the accurate basis vectors are used to construct the projectors.
- The P-system is solved using the least-squares approach, omitting the phase condition and the pseudo-arclength equation.
- The Q-systems are solved using two Picard iteration steps to compute Δq_r and Δq_γ with a zero starting value and we set $\Delta q_T = 0$. (This corresponds to $NPGS(2)$ and $CNP(2)$ of [15].)
- The Newton–Picard iterations were stopped once the residual and the updates Δp, ΔT and $\Delta\gamma$ were smaller than 10^{-6}.
- After the Newton–Picard iterations, we computed all Floquet multipliers larger than 0.7 until the corresponding basis satisfied $\sigma_{max}(Z_k) \leq 10^{-4}$. We also required that the changes between corresponding eigenvalues in two successive subspace iteration steps were smaller than 10^{-4}.

The results are reported in Table 1. Despite the small problem size, the computational cost is substantially lower when a Newton-Picard approach is used instead of a (Chord-)Newton method. Note that in [15] we reported lower numbers for the number of IVP-solves to compute these branches. However, the results in [15] were obtained with a less robust implementation, with highly optimized values of the parameters of the Newton–Picard method for each run.

We did observe some failures during our tests with the $NPGS$ and CNP method, most of which were caused by the omission of the phase condition. The least-squares based solution technique cannot remove a phase shift introduced by the predictor while phase conditions using the previous orbit as a reference solution can do that. In some cases, the phase shift between two successive orbits grew as the continuation proceeded and the predictor generated bad starting values, causing a failure of the continu-

ation procedure. Hence omitting the explicit phase condition decreases the robustness and is not as interesting as one would expect from the discussion in section 7.

We also computed a branch of periodic solutions for the two-dimensional variant of (8.1):

$$(8.3)\quad \begin{cases} \dfrac{\partial X}{\partial t} &= \dfrac{D_X}{L^2}\left(\dfrac{\partial^2 X}{\partial x^2}+\dfrac{\partial^2 X}{\partial y^2}\right)+X^2Y-(B+1)X+A, \\ \dfrac{\partial Y}{\partial t} &= \dfrac{D_Y}{L^2}\left(\dfrac{\partial^2 Y}{\partial x^2}+\dfrac{\partial^2 Y}{\partial y^2}\right)-X^2Y+BX \end{cases}$$

on the unit square $[0,1]\times[0,1]$ with the Dirichlet boundary conditions (8.2) on all boundaries. We used a second order spatial finite difference discretization with 10×10 and 20×20 discretization points, resulting in ODE systems of dimension 200 and 800 respectively. The time integration was done using the trapezoidal rule. We used the parameter values $A=2$, $B=5.45$, $D_X=0.004$ and $D_Y=0.008$ as in [21]. The trivial steady-state solution $X\equiv A$ and $Y\equiv B/A$ has a first Hopf bifurcation at $L\approx 0.72$, leading to a branch of stable periodic solutions. We computed this stable branch from $L=0.78$ to $L\approx 3$.

Using the 20×20 grid, we computed 17 orbits on this branch using the $NPGS$ method. Initially, at low values of L, the code works efficiently, requiring around 200 IVP solves per orbit. For higher values of L however, a cluster of eigenvalues develops around -0.75. At the last computed point ($L=3.12$) more than 40 Floquet multipliers are close to -0.75. The basis V_p grows from 12 vectors at the start point to 60 vectors at the last point. The number of matrix–vector products grows excessively towards the end of the computed branch. In total, we needed 4295 time integrations (an average of 253) to compute the orbits. However, 9848 matrix–vector products had to be computed to compute the dominant Floquet multipliers with the desired accuracy. Still, the code did not fail in the presence of a large cluster of eigenvalues and managed to increase its basis correctly as the computations proceeded.

However, the numerical results are probably not a correct representation of the infinite-dimensional system. Indeed, for these values of D_X and D_Y and for large L, the solution develops strong spatial gradients near the boundaries during part of the period and we can not capture the gradients well enough with the rather coarse discretization. We suspect that the inaccurate representation of the physical model causes the cluster of eigenvalues and the cluster might disappear on a finer grid.

8.2. Model of Elezgaray and Arneodo. We also computed a branch of a reaction-diffusion model studied by Elezgaray and Arneodo

in [3]:

$$\begin{cases} \dfrac{\partial u}{\partial t} = D\dfrac{\partial^2 u}{\partial^2 x} + \dfrac{1}{\epsilon}(v - (u^2 + u^3)), \\ \dfrac{\partial v}{\partial t} = D\dfrac{\partial^2 v}{\partial^2 x} + \alpha - u, \end{cases} \tag{8.4}$$

with Dirichlet boundary conditions

$$\begin{cases} u(0,t) \equiv u(1,t) \equiv -2, \\ v(0,t) \equiv v(1,t) \equiv -4. \end{cases} \tag{8.5}$$

D is used as the bifurcation parameter. ϵ and α are both fixed at 0.1. We used second order finite differences for the space discretization, the trapezoidal rule for time integration and variational equations for the matrix–vector products. The bifurcation diagram can be found in [6], with a different scaling of the equations however. (8.4)-(8.5) has a branch of periodic solutions that emanates from a steady-state Hopf bifurcation at $D \approx 0.02630$ and disappears in another steady-state Hopf bifurcation around $D \approx 0.03230$. On this branch, there are period doubling bifurcations around $D \approx 0.03208$ and $D \approx 0.03227$. In between the two period doubling bifurcations, the branch is unstable and there are various chaotic regimes limited by period doubling cascades near the two period doubling points. The solution develops strong spatial gradients and a very fine space discretization is needed to compute the branch accurately. We did computations using 63, 255 and 1023 discretization points and were able to compute the complete branch, including the unstable part in the chaotic region. All Floquet multipliers larger than 0.75 in modulus were also computed. In the region with chaotic dynamics, one of the Floquet multipliers grows to values around -190. Figure 4 shows the modulus of the computed dominant Floquet multipliers on part of the branch. 87 orbits were computed to construct Figure 4. There were 31 failures at which the stepsize was decreased. 13254 time integrations were required (9171 for successful continuation points and 4083 for failed points), or an average of 112.3 time integrations per successfully computed continuation point. We never needed more than 213 time integrations for a single orbit. Note that we used conservative settings and did not try to optimize the parameters.

9. Conclusions.

In this paper we have described a class of Newton-Picard methods for the efficient computation of periodic solutions of large-scale systems of ordinary differential equations. The methods are based on a single shooting approach and they combine a Newton iteration in a low-dimensional subspace (the eigenspace of the dominant Floquet multipliers) and a Picard iteration in the orthogonal complement. They are particularly efficient for systems that exhibit low-dimensional dynamics, e.g., for discretizations of

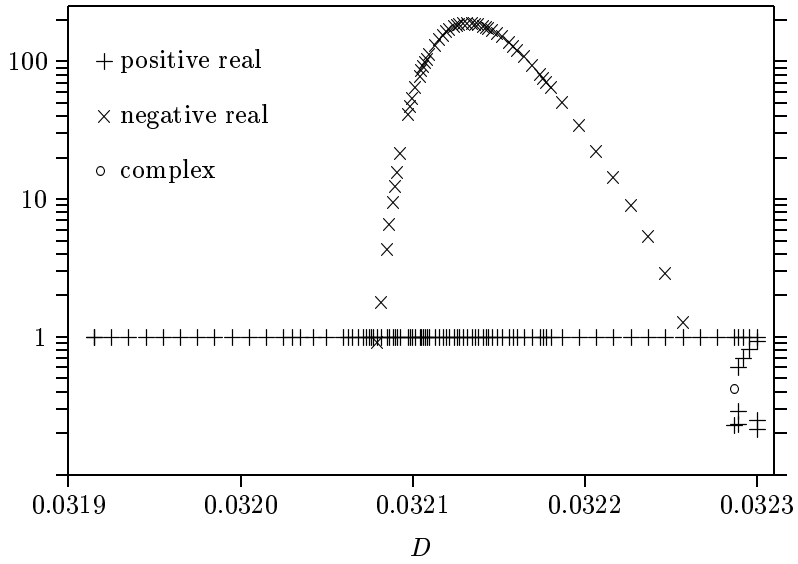

FIG. 4. *Modulus of the computed Floquet multipliers for the model of Elezgaray and Arneodo in terms of the parameter D, 1023 discretization points. Threshold ρ for the basis computation: 0.75. The pluses stand for positive real Floquet multipliers, the stars for negative real Floquet multipliers, and the circles stand for pairs of complex conjugate Floquet multipliers.*

partial differential equations, and when used in a continuation procedure. Since the dominant Floquet multipliers are computed, stability information is available and a bifurcation analysis can be performed.

The algebraic framework for Newton-Picard methods developed in this paper allows us to derive several variants and to analyze their convergence in detail. By exploiting the convergence properties within the implementation, we have achieved both efficiency and robustness. The paper also describes in detail several implementation aspects, such as the efficient and reliable computation of a basis for the low-dimensional subspace, the efficient implementation of the Picard step and the stable solution of the low-dimensional linear system. Finally, some numerical results are presented, which illustrate the efficiency of Newton-Picard methods for continuation and bifurcation analysis of periodic solutions, when compared with a classical ("full-Newton") shooting approach.

Several extensions to the Newton–Picard methods described in this paper have been developed. More sophisticated iteration schemes, such as Krylov iteration methods, can be used in the high-dimensional subspace instead of a Picard iteration [14]. The Newton–Picard approach has been extended to multiple shooting [14] and also to the solution of so-called extended or determining systems for bifurcation points [4]. Finally, Newton-Picard methods have been adapted to compute periodic solutions of delay

differential equations and their stability [16].

Acknowledgements. Kurt Lust is a postdoctoral associate of the Institute for Mathematics and its Applications, University of Minnesota. The authors gratefully acknowledge the financial support of the Fund for Scientific Research – Flanders for the research assistantship of Kurt Lust (92–97) and for project G.01235.96. This research is also supported by the Belgian programme on Interuniversity Poles of Attraction (IUAP P4/2), initiated by the Belgian State – Prime Minister's Service – DWTC, and by the Research Fund of K.U.Leuven (OT/94/16). The scientific responsibility rests with its authors.

REFERENCES

[1] J. Dennis and R. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, vol. 16 of Classics in Applied Mathematics, SIAM, 1996.

[2] E. Doedel, *AUTO: Software for continuation and bifurcation problems in ordinary differential equations*, report Applied Mathematics, California Institute of Technology, Pasadena, USA, 1986.

[3] J. Elezgaray and A. Arneodo, *Crisis-induced intermittent bursting in reaction-diffusion chemical systems*, Physical Review Letters, 68 (1992), pp. 714–717.

[4] K. Engelborghs, K. Lust, and D. Roose, *A Newton-Picard method for accurate computation of period doubling bifurcation points of large-scale systems of ODEs*, TW report 251, Department of Computer Science, Katholieke Universiteit Leuven, Belgium, January 1997. Submitted to IMA Journal of Numerical Analysis.

[5] G. Golub and C. Van Loan, *Matrix computations*, vol. 3 of John Hopkins series in the mathematical sciences, John Hopkins University Press, Baltimore, 1983.

[6] M. Graham and I. Kevrekidis, *Alternative approaches to the Karhunen-Loève decomposition for model reduction and data analysis*, Comp. Chem. Eng., 20 (1996), p. 495.

[7] B. Hassard, N. Kazarinoff, and Y. Wan, *Theory and applications of Hopf bifurcation*, Cambridge University Press, Cambridge, 1981.

[8] M. Holodniok, P. Knedlik, and M. Kubiček, *Continuation of periodic solutions in parabolic differential equations*, in Bifurcation: Analysis, Algorithms, Applications, T. Küpper, R. Seydel, and H. Troger, eds., ISNM 79, Birkhäuser, Basel, 1987, pp. 122–130.

[9] H. Jarausch and W. Mackens, *Numerical treatment of bifurcation branches by adaptive condensation*, in Numerical Methods for Bifurcation Problems, T. Küpper, H. D. Mittelmann, and H. Weber, eds., vol. 70 of ISNM, Birkhäuser, Basel, 1984, pp. 296–309.

[10] ———, *Computing bifurcation diagrams for large nonlinear variational problems*, in Large Scale Scientific Computing, P. Deuflhard and B. Enguist, eds., vol. 7 of Progress in Scientific Computing, Birkhäuser Verlag, 1987.

[11] ———, *Solving large nonlinear systems of equations by an adaptive condensation process*, Numer. Math., 50 (1987), pp. 633–653.

[12] A. Jennings and W. Stewart, *A simultaneous iteration algorithm for real matrices*, ACM Trans. of math. software, 7 (1981), pp. 184–198.

[13] Y. Kuznetsov and V. Levitin, *CONTENT, a multiplatform continuation environment*, technical report, CWI, Amsterdam, The Netherlands, 1996.

[14] K. Lust, *Numerical bifurcation analysis of periodic solutions of partial differential equations*, PhD thesis, Katholieke Universiteit Leuven, 1997.

[15] 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., (1996). Accepted for publication.

[16] T. Luzyanina, K. Engelborghs, K. Lust, and D. Roose, *Computation, continuation and bifurcation analysis of periodic solutions of delay differential equations*, International Journal of Bifurcation and Chaos, (1997). Accepted.

[17] K. Meerbergen and D. Roose, *Matrix transformations for computing rightmost eigenvalues of large sparse non-symmetric eigenvalue problems*, IMA Journal of Numerical Analysis, 16 (1996), pp. 297–346.

[18] K. Radhakrishnan and A. Hindmarsh, *Description and use of LSODE, the Livermore Solver for Ordinary Differential Equations*, LLNL Report UCRL-ID-113855, Lawrence Livermore National Laboratory, 1994. NASA Reference Publication 1327.

[19] W. Rheinboldt, *Numerical analysis of parametrized nonlinear equations*, vol. 7 of University of Arkansas Lecture Notes in the Mathematical Sciences, Wiley-Interscience Publication, New York, 1986.

[20] 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.

[21] D. Roose and S. Vandewalle, *Efficient parallel computation of periodic solutions of parabolic partial differential equations*, in Bifurcation and Chaos: Analysis, Algorithms, Applications, R. Seydel, F. W. Schneider, T. Küpper, and H. Troger, eds., vol. 97 of ISNM, Birkhäuser, Basel, 1991, pp. 307–317.

[22] Y. Saad, *Numerical methods for large eigenvalue problems*, Algorithms and architectures for advanced scientific computing, Manchester University Press, Manchester, 1992.

[23] R. Seydel, *Practical Bifurcation and Stability Analysis. From Equilibrium to Chaos*, Springer-Verlag, New York, second ed., 1994.

[24] G. Shroff and H. Keller, *Stabilization of unstable procedures: the recursive projection method*, SIAM J. Numer. Anal., 30 (1993), pp. 1099–1120.

[25] G. Stewart, *Simultaneous iteration for computing invariant subspaces of non-hermitian matrices*, Numer. Math., 25 (1976), pp. 123–136.