An Adaptive Newton–Picard Algorithm with Subspace Iteration for Computing Periodic Solutions

Kurt Lust, Dirk Roose, Alastair Spence, Alan Champneys

AN ADAPTIVE NEWTON–PICARD ALGORITHM WITH SUBSPACE ITERATION FOR COMPUTING PERIODIC SOLUTIONS*

K. LUST†, D. ROOSE†, A. SPENCE‡, AND A. R. CHAMPNEYS§

Abstract. This paper is concerned with the efficient computation of periodic orbits in large-scale dynamical systems that arise after spatial discretization of partial differential equations (PDEs). A hybrid Newton–Picard scheme based on the shooting method is derived, which in its simplest form is the recursive projection method (RPM) of Shroff and Keller [SIAM J. Numer. Anal., 30 (1993), pp. 1099–1120] and is used to compute and determine the stability of both stable and unstable periodic orbits. The number of time integrations needed to obtain a solution is shown to be determined only by the system's dynamics. This contrasts with traditional approaches based on Newton's method, for which the number of time integrations grows with the order of the spatial discretization. Two test examples are given to show the performance of the methods and to illustrate various theoretical points.

1. Introduction.

1.1. Continuation of periodic solutions. This paper is concerned with the efficient computation of periodic orbits of PDEs and the determination of their stability. Specifically we consider the autonomous dynamical system

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

with N "large," and with f derived from a finite element or finite difference spatial discretization of a parabolic PDE (see section 4). We will assume that f is C^2-continuous in x and λ in the region of interest. For a fixed λ, 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

$$\begin{aligned} x(T) - x(0) &= 0, \\ s(x(0),T) &= 0, \end{aligned} \tag{1.2}$$

where the second equation is a phase condition needed to eliminate the translational invariance of periodic solutions of autonomous dynamical systems (see section 4 and

The authors acknowledge the financial support of the British Council and Vlaamse Interuniversitaire Raad within the "British-Flemish Academic Research Collaboration Programme" and the Volkswagen Stiftung. This work was also supported by the Belgian programme on Interuniversity Poles of Attraction (IUAP 17), initiated by the Belgian State – Prime Minister's Service – DWTC. The scientific responsibility is assumed by its authors.

†Department of Computer Science, K.U. Leuven, Celestijnenlaan 200A, B-3001 Heverlee-Leuven, Belgium (kurt.lust@cs.kuleuven.ac.be, dirk.roose@cs.kuleuven.ac.be). K. Lust is Research Assistant of the National Fund for Scientific Research (Belgium).

‡School of Mathematics, University of Bath, Claverton Down, BA2 7AY Bath, UK (a.spence@maths.bath.ac.uk).

§Department of Engineering Mathematics, University of Bristol, Queen's Building, University Walk, BS8 1TI Bristol, UK (a.r.champneys@bristol.ac.uk). The research of this author was supported by a research assistantship from the SERC (UK) at the University of Bath.

[19, p. 251] for examples). When computing a branch of periodic solutions, the parameter λ is allowed to vary and an additional condition, the parameterizing equation, is added. The unknowns $x(0)$, T, and λ are found by solving

$$\begin{cases} x(T) - x(0) = 0, \\ s(x(0), T, \lambda) = 0, \\ n(x(0), T, \lambda; \tau) = 0 \end{cases} \tag{1.3}$$

with τ an artificial parameter. With a good choice for n, one is able to pass round turning points and detect other bifurcations without any difficulties. In our examples in section 4, we use pseudo-arclength parameterization [2], but other choices are possible [16, 19].

1.2. Some numerical methods. When N is small a common approach is to rescale time to obtain a new boundary value problem of period 1 where T is now an unknown parameter. This new problem can then be solved by a direct method for a two-point boundary value problem. The stability of the orbit is determined by its Floquet multipliers, the eigenvalues of the monodromy matrix (see (1.5)). If care is taken, this matrix can be recovered from the direct method (see [2, 4]).

In the problems of interest here, with N large and the Jacobian of f sparse, a direct approach is likely to prove infeasibly expensive. For this reason we shall use a shooting approach (though this approach has its limitations as discussed in [19]). For a given initial condition $x(0)$, we denote the solution of (1.1) at $t = T$ by $\varphi(x(0), T, \lambda)$. System (1.2) can now be written as

$$\begin{array}{rcl} r(x(0), T, \lambda) & := & \varphi(x(0), T, \lambda) - x(0) = 0, \\ s(x(0), T, \lambda) & = & 0, \\ n(x(0), T, \lambda) & = & 0, \end{array} \tag{1.4}$$

and we denote a solution of (1.4) by $(x^*(0), T^*, \lambda^*)$. Note we do not make the rescaling of time mentioned above. If $\frac{\partial \varphi}{\partial x}$ denotes the partial derivative of $\varphi(x(0), T, \lambda)$ with respect to $x(0)$, then the monodromy matrix M^* is given by

$$M^* := \left. \frac{\partial \varphi}{\partial x} \right|_{(x^*(0), T^*, \lambda^*)}. \tag{1.5}$$

We denote the eigenvalues (Floquet multipliers) of M^* by μ_i^*, $i = 1, \ldots, N$. As is well known (see, for example, [19]), one eigenvalue of M^* has the value 1. The corresponding eigenvector is the tangent vector to the limit cycle in $(x(0)^*, \lambda^*)$,

$$b^* := \left. \frac{\partial \varphi}{\partial T} \right|_{(x(0)^*, T^*, \lambda^*)} = f(\varphi(x(0)^*, T^*, \lambda^*), \lambda^*). \tag{1.6}$$

If any eigenvalue lies outside the unit circle, the periodic orbit is unstable.

To compute *stable* periodic orbits a natural way requires merely the integration of (1.1) over a sufficiently long time interval. If $x(0)$ is close enough to the limit cycle then the trajectory will converge to the periodic solution and one can read off the value for the unknown period T^*. This is essentially a form of Picard iteration (see [17]). Indeed if the period T^* were known exactly, one would carry out Picard iteration on a Poincaré map to find a stable periodic orbit. Variations on this theme are possible and they are well suited to problems where N is large, but they break down or have very slow convergence if M^* has eigenvalues outside or near the unit circle

(apart from the trivial one at 1). A Newton-like method applied to (1.4) directly would be able to compute both stable and unstable orbits but would involve the computation or approximation of $\frac{\partial \varphi}{\partial x}$ either by numerical differentiation or by solving the variational form of (1.1). In the former case, the nonlinear system (1.1) would have to be integrated N times with perturbed initial data, and in the latter case an N^2-dimensional linear initial value problem (IVP) has to be integrated together with (1.1). Thus both approaches are prohibitively expensive when N is large. A Newton-like method also requires the expensive storage and factorization of the full $N \times N$ matrix.

This paper is concerned with the efficient solution of (1.4) and the computation of the stability-determining eigenvalues of M^*. Of primary concern is the desire to avoid the explicit calculation of M^*. The approach used here requires only the calculation of the action of M^* on a p-dimensional subspace of $\mathbb{R}^N$ with $p \ll N$. As we show in section 2, this is a computationally less expensive task. To solve (1.4) we shall use a hybrid Newton–Picard approach which was motivated by and is an extension of the RPM of Shroff and Keller [20]. However, though the philosophy is the same, there are several differences in the implementation forced on us by the nature of M^* and by the fact that we seek $x(0)$, λ, *and* T rather than just $x(0)$, λ. Our approach is influenced by the knowledge that the problem is set in a continuation framework, and hence, for a given λ, reasonably accurate starting approximations for $x(0)$ and T will be known, and a rough idea of the distribution of the Floquet multipliers will also be known. This provides us with some degree of confidence in some of the linear algebra techniques which would not be satisfied in a one-off problem.

The present paper is an extension of the preliminary results presented in [17] which concerned only the basic idea of a Newton–Picard algorithm for periodic orbits. The plan of the paper is as follows. In section 2 we derive from first principles several approximate Newton methods for the computation of both equilibrium points and periodic solutions. In so doing we recover the RPM of [20] but also produce other methods which prove superior in the examples discussed in section 4. The relation of these approximate Newton methods to earlier work is also discussed in section 2, as are their convergence properties. In section 3, we discuss the efficient implementation of the methods. This includes iterative methods to compute a maximal dominant invariant subspace of a matrix using only matrix–vector products and not requiring the explicit computation of the matrix. Some other important implementation details are also discussed in this section. In section 4 we present some numerical experiments which confirm the theoretical results and illustrate the effectiveness of our approach. Finally, section 5 presents our conclusions.

2. Newton–Picard methods.

2.1. Derivation.

Consider the solution of the nonlinear system

$$r(x) := F(x) - x = 0, \quad x \in \mathbb{R}^N, \quad F : \mathbb{R}^N \mapsto \mathbb{R}^N, \tag{2.1}$$

where $F(x)$ is at least twice differentiable in the neighborhood of x^* a locally unique root of (2.1). It is well known that under certain conditions, the simple Picard iteration

$$x^{(\nu+1)} = F(x^{(\nu)}), \quad \nu = 0, 1, 2, \ldots \tag{2.2}$$

converges linearly if $r_\sigma[F_x(x^*)] < 1$. (Here we use $r_\sigma[\cdot]$ to denote the spectral radius of a matrix, which is equal to the modulus of the eigenvalue of largest modulus.) In

contrast, (2.2) typically diverges if $F_x(x^*)$ has an eigenvalue outside the unit circle. Newton's method,

$$\left\{ \begin{array}{l} \left[F_x(x^{(\nu)}) - I\right] \Delta x^{(\nu)} = -\left(F(x^{(\nu)}) - x^{(\nu)}\right), \\ x^{(\nu+1)} = x^{(\nu)} + \Delta x^{(\nu)}, \quad \nu = 0,\, 1,\, 2,\, \ldots \end{array} \right. \tag{2.3}$$

shows quadratic convergence in a neighborhood of x^* if $F_x(x^*) - I$ is nonsingular, but it needs the construction and factorization of one or more $N \times N$ matrices. From now on, we will drop the superscript (ν) from our notation whenever the formula remains clear. One can easily extend Newton's scheme to compute a solution to the parameter dependent system

$$\left\{ \begin{array}{ll} r(x,\lambda) = F(x,\lambda) - x = 0, & F : \mathbb{R}^N \times \mathbb{R} \mapsto \mathbb{R}^N, \\ n(x,\lambda) = 0, & n : \mathbb{R}^N \times \mathbb{R} \mapsto \mathbb{R}, \end{array} \right. \tag{2.4}$$

where $n(x,\lambda)$ is some (pseudo-arclength) parameterization, and to (1.4)

$$\left\{ \begin{array}{l} r(x(0),T,\lambda) = \varphi(x(0),T,\lambda) - x(0) = 0, \\ s(x(0),T,\lambda) = 0, \\ n(x(0),T,\lambda) = 0, \end{array} \right. \tag{2.5}$$

where it is assumed φ, s, and n are twice differentiable with respect to $x(0)$, T, and λ. For φ, this follows from the C^2-continuity of $f(x,\lambda)$ in (1.1) [1].

For (2.5), the use of Newton's method leads to the repetitive solution of linear systems in the form

$$\left[\begin{array}{ccc} M - I & b_1 & b_2 \\ c_1^T & d_{1\,1} & d_{1\,2} \\ c_2^T & d_{2\,1} & d_{2\,2} \end{array} \right] \left[\begin{array}{c} \Delta x(0) \\ \Delta T \\ \Delta \lambda \end{array} \right] = - \left[\begin{array}{c} r(x(0),T,\lambda) \\ s(x(0),T,\lambda) \\ n(x(0),T,\lambda) \end{array} \right], \tag{2.6}$$

where

$$\left[\begin{array}{ccc} M - I & b_1 & b_2 \\ c_1^T & d_{1\,1} & d_{1\,2} \\ c_2^T & d_{2\,1} & d_{2\,2} \end{array} \right] = \frac{\partial(r,s,n)}{\partial(x(0),T,\lambda)}.$$

For (2.1) and (2.4), Jarausch and Mackens [8, 9, 10] and Shroff and Keller [20] both had the idea to use Newton's method on a small-dimensional subspace, where the Picard iteration (time integration in case of (2.5)) is slowly convergent or unstable, and to use the Picard iteration on the complement of that subspace. Our approach to the efficient solution of periodic orbits of (1.1) via the solution of (2.5) is motivated by the treatment in sections 2 and 7 of [20]. We make the following assumptions.

ASSUMPTION 2.1. *Let $y^* = (x(0)^*, T^*, \lambda^*)$ denote an isolated solution to* (2.5)*, and let $\mathcal{B}$ be a small neighborhood of y^*. Let $M(y) = \frac{\partial \varphi}{\partial x}(y)$ 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*

$$C_\rho = \{|z| < \rho\}, \quad 0 < \rho < 1 \tag{2.7}$$

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 this assumption is more restrictive than that in [20]. At y^*, M^* is the monodromy matrix. Our method is designed to be efficient for $p \ll N$. Let $V_p \in \mathbb{R}^{N\times p}$ be a basis for the subspace $\mathcal{U}$ of $\mathbb{R}^N$ spanned by the (generalized) eigenvectors of $M(x(0), T, \lambda)$ corresponding to the eigenvalues μ_i, $i = 1, \ldots, p$ and $V_q \in \mathbb{R}^{N\times(N-p)} = \mathbb{R}^{N\times q}$ a 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 $M(x(0), T, \lambda)$. V_p and V_q can be computed by the real Schur factorization of $M(x(0), T, \lambda)$. Note that 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, Assumption 2.1 is observed in practice and so is not unreasonable.

We can construct orthogonal projectors P and Q of $\mathbb{R}^N$ onto $\mathcal{U}$ and $\mathcal{U}^\perp$, respectively, as

$$\begin{aligned} P &:= V_p V_p^T, \\ Q &:= V_q V_q^T = I_N - V_p V_p^T. \end{aligned} \tag{2.8}$$

For any $x(0) \in \mathbb{R}^N$ there is the unique decomposition

$$x(0) = V_p\bar{p} + V_q\bar{q} = p + q, \quad p = V_p\bar{p} = Px(0), \quad q = V_q\bar{q} = Qx(0) \tag{2.9}$$

with $\bar{p} \in \mathbb{R}^p$ and $\bar{q} \in \mathbb{R}^{N-p}$. Substituting (2.9) in (2.6) and multiplying the first N equations at the left-hand side with $[\ V_q \quad V_p\ ]^T$, one obtains

(2.10)

$$\begin{bmatrix} V_q^T(M-I)V_q & 0 & V_q^Tb_1 & V_q^Tb_2 \\ V_p^TMV_q & V_p^T(M-I)V_p & V_p^Tb_1 & V_p^Tb_2 \\ c_1^TV_q & c_1^TV_p & d_{1\,1} & d_{1\,2} \\ c_2^TV_q & c_2^TV_p & d_{2\,1} & d_{2\,2} \end{bmatrix} \begin{bmatrix} \Delta\bar{q} \\ \Delta\bar{p} \\ \Delta T \\ \Delta\lambda \end{bmatrix} = - \begin{bmatrix} V_q^Tr \\ V_p^Tr \\ s \\ n \end{bmatrix}.$$

Here we have used $V_p^TV_q = 0_{p\times q}$, $V_q^TV_p = 0_{q\times p}$, and $V_q^TMV_p = 0_{q\times p}$ with the latter equality holding since $\mathcal{U}$ is an invariant subspace of $M(x(0), T, \lambda)$. Remark that $b_1(x(0)^*, T^*, \lambda^*)$ is an eigenvector of $M(x(0)^*, T^*, \lambda^*)$ for the eigenvalue 1, and thus by Assumption 2.1 a vector in the subspace $\mathcal{U}$ at $(x(0)^*, T^*, \lambda^*)$. Therefore, $V_q^Tb_1$ converges to zero as the iteration converges. We will put this term to zero.

Let us first consider the case where λ is kept fixed. Equation (2.10) with the above approximation for $V_q^Tb_1$ then reduces to the block triangular system

$$\begin{bmatrix} V_q^TMV_q - I_q & 0 & 0 \\ V_p^TMV_q & V_p^TMV_p - I_p & V_p^Tb_1 \\ c_1^TV_q & c_1^TV_p & d_{1\,1} \end{bmatrix} \begin{bmatrix} \Delta\bar{q} \\ \Delta\bar{p} \\ \Delta T \end{bmatrix} = - \begin{bmatrix} V_q^Tr \\ V_p^Tr \\ s \end{bmatrix}. \tag{2.11}$$

This is no real saving since we want to avoid the computation of the large matrix $V_q^TMV_q$. However, we can approximate the solution to the first q equations of (2.11) by the Picard scheme

$$\begin{cases} \Delta\bar{q}^{[0]} = 0, \\ \Delta\bar{q}^{[i]} = V_q^TMV_q\Delta\bar{q}^{[i-1]} + V_q^Tr, \quad i = 1, \ldots, l, \\ \Delta\bar{q} = \Delta\bar{q}^{[l]} = \sum_{i=0}^{l-1}(V_q^TMV_q)^iV_q^Tr. \end{cases} \tag{2.12}$$

By construction, $r_\sigma[V_q^T M V_q] = |\mu_{p+1}| < \rho < 1$. Thus the linear Picard scheme (2.12) converges asymptotically. Obviously (2.12) corresponds to approximately solving $(V_q^T M V_q - I_q)\Delta\bar{q} = -V_q^T r$ by multiplying the right-hand side by the l terms Neumann series for $(V_q^T M V_q - I_q)^{-1}$. $\Delta\bar{p}$ and ΔT can then be computed from

$$\text{(2.13)} \quad \begin{bmatrix} V_p^T M V_p - I_p & V_p^T b_1 \\ c_1^T V_p & d_{1\,1} \end{bmatrix} \begin{bmatrix} \Delta\bar{p} \\ \Delta T \end{bmatrix} = - \begin{bmatrix} V_p^T r \\ s \end{bmatrix} - \begin{bmatrix} V_p^T M V_q \Delta\bar{q} \\ c_1^T V_q \Delta\bar{q} \end{bmatrix}.$$

Except for highly nonnormal cases, (2.12) with $l = 1$ is enough to make the overall scheme convergent for a starting value close enough to $(x(0)^*, T^*, \lambda^*)$ (see section 2.3).

Using Taylor's theorem, the right-hand side of (2.13) can be transformed to

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T b_1 \\ c_1^T V_p & d_{1\,1} \end{bmatrix} \begin{bmatrix} \Delta\bar{p} \\ \Delta T \end{bmatrix} = - \begin{bmatrix} V_p^T r(x(0) + V_q\Delta\bar{q}, T) \\ s(x(0) + V_q\Delta\bar{q}, T) \end{bmatrix} + \text{h.o.t.}$$

This is seen to be a *Gauss–Seidel approach* since we first solve an approximate version of the equations for $\Delta\bar{q}$ and then use the updated $\bar{q}$ to compute $\Delta\bar{p}$ and ΔT. The variant based on l steps of the linear Picard scheme in (2.12) will be denoted by *NPGS(l)*. *Jacobi-type variants* are derived by putting the terms $V_p^T M V_q$ and $c_1^T V_q$ in (2.11) to 0, although these terms can (and often will) be large. We will further denote this variant by *NPJ(l)*. Remark that our NPJ(1) scheme applied to the equilibrium point problem (2.1) is the method presented in [20, section 2]. When F_x in (2.1) is normal, the NPJ(l) and NPGS(l) variants are equivalent. The NPGS(l) method can formally be written as an approximate Newton method in the form

(2.14)

$$\begin{bmatrix} -\left(\sum_{i=0}^{l-1}(V_q^T M V_q)^i\right)^{-1} & 0 & 0 \\ V_p^T M V_q & V_p^T M V_p - I_p & V_p^T b_1 \\ c_1^T V_q & c_1^T V_p & d_{1\,1} \end{bmatrix} \begin{bmatrix} \Delta\bar{q} \\ \Delta\bar{p} \\ \Delta T \end{bmatrix} = - \begin{bmatrix} V_q^T r \\ V_p^T r \\ s \end{bmatrix}.$$

The NPJ(l) Jacobi variant is obtained by putting the $(2,1)$ and $(3,1)$ blocks to zero. This form will be used in the convergence study.

Let us now reintroduce the parameter λ. In (2.10), there is no reason for the term $V_q^T b_2$ to be small, and so the system (2.10) is fully coupled. However, the system matrix (with $V_q^T b_1$ set to zero) is just a rank-one update of the block triangular matrix

$$\text{(2.15)} \qquad B = \begin{bmatrix} V_q^T M V_q - I_q & 0 & 0 & 0 \\ V_p^T M V_q & V_p^T M V_p - I_p & V_p^T b_1 & V_p^T b_2 \\ c_1^T V_q & c_1^T V_p & d_{1\,1} & d_{1\,2} \\ c_2^T V_q & c_2^T V_p & d_{2\,1} & d_{2\,2} \end{bmatrix}.$$

We solve this system using the Sherman–Morrison formula (see, e.g., [15, 16]). Let

$$\theta = \begin{bmatrix} V_q^T b_2 \\ 0_p \\ 0 \\ 0 \end{bmatrix}, \quad \xi = \begin{bmatrix} 0_q \\ 0_p \\ 0 \\ 1 \end{bmatrix}, \quad \Delta u = \begin{bmatrix} \Delta\bar{q} \\ \Delta\bar{p} \\ \Delta T \\ \Delta\lambda \end{bmatrix}, \quad t = - \begin{bmatrix} V_q^T r \\ V_p^T r \\ s \\ n \end{bmatrix}.$$

Provided B is regular, a solution to

$$(B + \theta\xi^T)\Delta u = t$$

can be obtained by solving the two systems

$$Bw = \theta, \quad Bv = t. \tag{2.16}$$

We can compute $\Delta\lambda$ from

$$\Delta\lambda = \xi^T \Delta u = \frac{\xi^T v}{1 + \xi^T w} \tag{2.17}$$

and the other unknown from

$$\Delta u = v - \Delta\lambda\, w. \tag{2.18}$$

If $B + \theta\xi^T$ is regular then

$$1 + \xi^T w = 1 + w_{N+2} \neq 0.$$

Indeed,

$$B^{-1}(B + \theta\xi^T) = I + w\xi^T = \begin{bmatrix} 1 & 0 & \cdots & 0 & w_1 \\ 0 & 1 & \cdots & 0 & w_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & w_{N+1} \\ 0 & 0 & \cdots & 0 & 1 + w_{N+2} \end{bmatrix}$$

so $1+\xi^T w = 0$ would imply that $I+w\xi^T$ and thus $B+\theta\xi^T$ is singular. The two systems (2.16) can be solved (approximately) as before by first solving for $\Delta\bar{q}$ using (2.12) and then solving the remaining equations for the other unknowns. Since this approach is relevant in a continuation context using Newton–Picard iterations we denote this variant by CNP(l) (i.e., continuation Newton–Picard). Other variants can be derived by deleting the $(2,1)$, $(3,1)$, $(4,1)$, and/or the $(1,4)$ blocks. In all cases, the resulting system is partially or fully decoupled and only one Q-system needs to be solved at each step. When applying the method to the continuation of steady-state solutions, we obtain the scheme of [20, section 7] by choosing $l = 1$ and neglecting the $V_p^T M V_q$ and $c_2^T V_q$ terms. The CNP(l) method seems the most expensive variant, requiring the solution of two Q-systems at each iteration step. However, the CNP(l) method proved to be more robust in our tests.

2.2. Comparison with other work. Recursive projection methods were first proposed by Jarausch and Mackens [8, 9, 10][1] for solving

$$Au = f(u), \tag{2.19}$$

where A is a symmetric positive definite matrix and $\frac{\partial f(x)}{\partial x}$ is symmetric. For $A = I$, our NPJ(1) and NPGS(1) schemes reduce to their method. Based on a decoupling lemma (stating that a change in p has no influence on the Q-projection of the residual and vice versa) and a trust-region approach, a criterion to decide whether to do a (damped) Q-step, a (damped) P-step or an update of the basis, is derived. Under certain conditions, this results in a globally convergent algorithm. For continuation, they use the Moore–Penrose inverse to solve the underdetermined P-system and correct

[1] Jarausch and Mackens call the method "condensed Newton–supported Picard."

q after each update to eliminate the influence of a change of λ on the Q-projection of the residual. Our case is more difficult because of the (sometimes large) influence of the Q-step on the P-projection of the residual. This results in a block triangular system when parameters are kept fixed and a fully coupled system when a parameter is varied. There is no straightforward extension of the decoupling lemma to unsymmetric systems when using one set of orthogonal projectors. Therefore, in this paper we always do a P- and a Q-step together. In section 3.2 we will derive a criterion for the basis update.

In later work, Jarausch developed alternatives for nonsymmetric systems. Two schemes for finding equilibrium points of parabolic PDEs are developed in [6]. The partially decoupled case is similar to our NPGS(1) scheme, while the fully decoupled scheme uses oblique projectors based on both left and right invariant subspaces. The splitting is not based on the convergence properties of a Picard scheme but on the physically relevant eigenvalues with respect to stability: the projectors are based on subspaces corresponding to eigenvalues which have a real part larger than a certain (negative) threshold. Several algorithms to solve the Q-system are proposed.

Computing the subspaces needed in the algorithms in [6] without constructing the matrix explicitly is much harder than computing the subspaces needed in our algorithms or in the algorithm in [8, 9, 10]. A complete decoupling has certain advantages, e.g., in bifurcation analysis, but the oblique projectors needed in [6] to achieve this impose numerical difficulties. As a solution to these problems, Jarausch developed another method to solve systems of the form (2.1) in [7]. A completely decoupled system is derived using orthogonal projectors and a rotator with the result that singular subspaces of the Jacobian of f are used, instead of invariant subspaces.

In [20], Shroff and Keller also generalized the method of [8, 9, 10]. They look at the method as a way to stabilize a Picard iteration scheme. Their method is completely equivalent to our NPJ(1) scheme. However, by thinking of the procedure as an approximate Newton method, we have derived some other variants with better convergence properties; see section 4. In some of those variants, the influence of the Q-updates on the P-projection of the residual shows up, so these schemes should be more robust.

2.3. Convergence discussion. A convergence analysis for RPM under rather restrictive assumptions is given in [20]. For our application a full convergence analysis would require very careful discussion (including an account of norms of nonnormal matrices, etc.), which would probably have limited practical value. In this section we present two results on asymptotic convergence on which a more complete convergence account could be based.

We concentrate on methods for the solution of (2.1) and refer to the extension to (2.5) in the last paragraph. A fixed point method

$$x^{(\nu+1)} = G(x^{(\nu)}) \tag{2.20}$$

is asymptotically convergent if

$$r_\sigma[G_x(x^*)] < 1 \tag{2.21}$$

and the asymptotic convergence rate is $r_\sigma[G_x(x^*)]$. Corresponding to Assumption 2.1, we suppose the following assumption.

ASSUMPTION 2.2. *Let x^* denote an isolated solution to* (2.1) *and let $\mathcal{B}$ be a small neighborhood of x^*. Let $M(x) = \frac{\partial F}{\partial x}(x)$ for $x \in \mathcal{B}$ and denote its eigenvalues by*

μ_i, $i = 1, \ldots, N$. Assume that for all $x \in \mathcal{B}$ precisely p eigenvalues lie outside the disk

$$C_\rho = \{|z| < \rho\}, \quad 0 < \rho < 1 \tag{2.22}$$

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

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

The NPGS(l) and the NPJ(l) schemes can both be rewritten in the form

$$x^{(\nu+1)} = G(x^{(\nu)}) = x^{(\nu)} + H(x^{(\nu)})r(x^{(\nu)}),$$

where

$$H(x) = H_{GS}(x) = V_q \left(\sum_{i=0}^{l-1} (V_q^T M V_q)^i \right) V_q^T + V_p (I_p - V_p^T M V_p)^{-1} V_p^T \left(I + M V_q \left(\sum_{i=0}^{l-1} (V_q^T M V_q)^i \right) V_q^T \right)$$

and

$$H(x) = H_J(x) = V_q \left(\sum_{i=0}^{l-1} (V_q^T M V_q)^i \right) V_q^T + V_p (I_p - V_p^T M V_p)^{-1} V_p^T,$$

respectively. In these formulas V_p, V_q, and $M = F_x(x)$ depend on x.

The following lemma provides the spectral radius results for NPJ(l) and NPGS(l).

LEMMA 2.3. *Let x^* be an isolated solution of $r(x) = 0$. Suppose*

$$F(x) \text{ is twice differentiable} \tag{2.23}$$

and

$$1 \notin \sigma\{V_p^T F_x(x^*) V_p\}. \tag{2.24}$$

For both NPJ(l) and NPGS(l) schemes, the spectral radius of the Jacobian of G at x^* satisfies

$$r_\sigma[G_x(x^*)] < \rho^l < 1.$$

Proof. The following proof is valid for both $H = H_{GS}$ and $H = H_J$. Since $r(x^*) = 0$,

$$G_x(x^*) = I + H(x^*) r_x(x^*).$$

Let $V = [\ V_q \quad V_p\ ]$ with p defined as in Assumption 2.2; then

$$r_\sigma[G_x(x^*)] = r_\sigma[V^T G_x(x^*) V] = r_\sigma[I + (V^T H(x^*) V)(V^T r_x(x^*) V)]$$

and

$$V^T H(x^*)V = \begin{bmatrix} \sum_{i=0}^{l-1}(V_q^T M V_q)^i & 0 \\ * & (I_p - V_p^T M V_p)^{-1} \end{bmatrix},$$

$$V^T r_x(x^*)V = \begin{bmatrix} V_q^T M V_q - I_q & 0 \\ V_p^T M V_q & V_p^T M V_p - I_p \end{bmatrix},$$

and hence

$$V^T G_x(x^*)V = \begin{bmatrix} (V_q^T M V_q)^l & 0 \\ * & 0 \end{bmatrix},$$

where the values at the stars depend on the type of the scheme. It directly follows from Assumption 2.2 that

$$r_\sigma[G_x(x^*)] = |\mu_{p+1}|^l < \rho^l < 1. \qquad \square$$

COROLLARY 2.4. *Under the conditions of Lemma* 2.3,

1. *NPGS(l) and NPJ(l) schemes are asymptotically convergent in a neighborhood of the solution if* $\rho < 1$;
2. *the asymptotic convergence rate of NPGS(l) and NPJ(l) is* $|\mu_{p+1}|^l$.

For the computation of periodic solutions with shooting and nonvarying parameters, a similar convergence result can be proved under the conditions that $\varphi(x(0), T)$ and $s(x(0), T)$ are twice continuously differentiable with respect to $x(0)$ and T in a neighborhood of $(x(0)^*, T^*)$ and

$$1 \notin \sigma \left\{ \begin{bmatrix} V_p^T M(x(0)^*, T^*) V_p & V_p^T b_1(x(0)^*, T^*) \\ c_1^T(x(0)^*, T^*) V_p & d_{1\,1}(x(0)^*, T^*) \end{bmatrix} \right\}.$$

As long as 1 is a simple eigenvalue of $M(x(0)^*, T^*)$, the last condition can always be satisfied using a suitable phase condition. Although NPJ(l) and NPGS(l) methods have the same asymptotic convergence rate, NPGS(l) methods perform better as is indicated in section 4. There is no similar theorem for the continuation variants in the general case.

3. Computation of the projectors and other algorithmic details. In this section, we will explain some of the implementation issues. First, we will discuss how one can avoid the expensive computation of the basis V_q and how one can compute matrix–vector products Mv without first building M. In the second part of this section, we will discuss the computation of an (approximate) basis V_p.

3.1. Implementing the Newton–Picard steps. In the Newton part of the scheme (see, e.g., (2.13)) and for the update of $x(0)$, we do not need the q-dimensional vector $\Delta\bar{q}$ itself but only the N-dimensional vector $\Delta q = V_q \Delta\bar{q}$. Therefore, (2.12) can be rewritten as follows (using $Q = V_q V_q^T = I_N - V_p V_p^T$):

$$\begin{cases} \Delta q^{[0]} = 0, \\ \Delta q^{[i]} = QM\Delta q^{[i-1]} + Qr, \quad i = 1, \ldots, l, \\ \Delta q = \Delta q^{[l]}. \end{cases} \tag{3.1}$$

Next we need matrix–vector products Mv in (3.1), the coupling term $V_p^T M \Delta q$, and in the Newton system (the term MV_p). Since $M = \partial\varphi(x(0), T, \lambda)/\partial x(0)$, those

matrix–vector products are rescaled directional derivatives that can be computed using the variational equations or finite differences. Thus the explicit computation of the full monodromy matrix M is avoided. In the first approach we define the fundamental matrix $W(t)$ at $\overline{x}(t)$ by the variational equation

$$\dot{W}(t) = f_x(\overline{x}(t))W(t),\ W(0) = I_N. \tag{3.2}$$

It is easy to verify that if $\overline{x}(t) = \varphi(x(0), t, \lambda)$, then $M = W(T)$. Equation (3.2) represents a set of N^2 linear IVPs, and so it is impractical to compute M if N is large. However, in applying our algorithms, for a given x and T, we need only compute Mv for a small number of vectors $v \in \mathbb{R}^N$. From (3.2) it follows that

$$\dot{W}(t)v = f_x(\varphi(x(0), t, \lambda), \lambda)W(t)v, \quad W(0)v = v,$$

or, putting $z(t) = W(t)v$,

$$\dot{z} = f_x(\varphi(x(0), t, \lambda), \lambda)z, \quad z(0) = v, \tag{3.3}$$

with the result that $Mv = z(T)$. The second approach is to use a finite difference approximation to

$$Mv = \left.\frac{\partial \varphi}{\partial x}\right|_{(x(0),T,\lambda)} v$$

given by

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

When using the finite difference approximations, 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. Thus in both approaches the calculation of the action of M on a vector v requires the solution of one IVP of dimension N, assuming that $\phi(x(0), T, \lambda)$ has already been computed and that the trajectory has also been stored when the variational equations are used. In the latter case, (3.3) is a linear IVP. Similarly, to calculate the action of M on the p-dimensional space $\mathcal{U}$, p IVPs need to be solved.

3.2. Computation of the projectors. The most important aspect of the Newton–Picard algorithms is the efficient calculation and repeated updating of the low-dimensional subspace $\mathcal{U}$. Recall that $\mathcal{U}$ is defined to be the space spanned by the (generalized) eigenvectors of M corresponding to Floquet multipliers of modulus greater than ρ. Since we are only interested in the dominant eigenvalues and since we are able to compute matrix–vector products with the matrix M, it is natural to use subspace iteration to compute the dominant eigenvalues and eigenvectors. This is also proposed in [8, 9, 10] and [20]. In order to keep the number of (expensive) matrix–vector multiplications low, we will use the most sophisticated version of this algorithm, namely, subspace iteration with projection after each iteration and locking (deflation), as described by Saad in [18] (Algorithm 5.4 with iter = 1). Since the details of the locking process are quite technical and would obscure the essential idea of the method, we now outline the procedure, ignoring the locking process.

Let us first discuss the subspace iteration process, where we keep M fixed, and assume p is known. We will use p_e additional vectors to accelerate the convergence and to aid the detection of eigenvalues leaving C_ρ. Let

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

be an initial guess for an orthonormal basis for $\mathcal{U}$. We extend this basis to the orthonormal basis

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

where $v_{p+1}^{[0]}$, ..., $v_{p+p_e}^{[0]}$ are guesses for the next p_e dominant Schur vectors. We rearrange the order of operations in algorithm 5.3 in [18] to further reduce the number of matrix–vector operations and obtain the following algorithm.

ALGORITHM 3.1. *Subspace iteration with projection*

Input: $V_e^{[0]} = \left[\begin{array}{ccccc} v_1^{[0]} & \cdots & v_p^{[0]} & \cdots & v_{p+p_e}^{[0]} \end{array} \right]$
routine to compute Mv.

Output: V_e, *where span*$(V_e[1 \cdots p])$ *is a good approximation for* $\mathcal{U}$.

begin
  $V_e = V_e^{[0]}$
  repeat
    Compute $W = MV_e$.
    Compute $S = V_e^T M V_e = V_e^T W$.
    Compute the Schur vectors $Y = \left[\begin{array}{ccc} y_1 & \cdots & y_{p+p_e} \end{array} \right]$ *of* S, *order them according to decreasing modulus of the corresponding eigenvalue.*
    $V_e \leftarrow WY$.
    Orthonormalize V_e.
  until *convergence*
end

The matrix–vector products $MV_e[1 \cdots p]$ in the last iteration step of this algorithm are also needed to compute the terms $V_p^T M V_p$ in (2.14) and (2.15). The convergence properties for Algorithm 3.1 are proven in [18]. The convergence factor for a simple eigenvalue μ_i is $|\mu_{p+p_e+1}/\mu_i|$, where μ_{p+p_e+1} is the next most dominant eigenvalue of M. From this formula we see that the more dominant eigenvalues converge faster than the other ones. We exploit this feature by using the version of subspace iteration with projection and locking as described in Algorithm 5.4 in [18]. The convergence criterion is based on [21] and requires no new matrix–vector products with M. For small values of l and when a good starting value is available, one subspace iteration step per Newton–Picard step is usually sufficient. For large values of l and small values of p_e, or before the first Newton–Picard steps, more subspace iteration steps are needed. Locking then prevents updating almost converged vectors and is especially useful when some of the Floquet multipliers are large. Once the Newton–Picard procedure has sufficiently converged and some eigenvalues of M do not change from step to step, we lock those converged eigenvalues to save some computation time. We then gradually switch to lower-rank updates of the projected Jacobian matrix. However, this strategy should be used with care, since preliminary locking of vectors and failing to unlock them again can lead to slow convergence or divergence of the algorithm. All vectors should be unlocked to compute accurate values for the Floquet multipliers after the final Newton–Picard step.

When computing a branch of periodic solutions, the initial guess $V_e^{[0]}$ may be the final basis for the periodic solution at the previous continuation point. However, it is safer to add some random components to the vectors to be sure that new eigenvectors can enter the basis rapidly. For the first point on a branch we can either start with random vectors and do some subspace iterations without a Newton–Picard step 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 rightmost eigenvalues of f_x. These eigenvalues and the corresponding basis can be computed using appropriate iterative techniques; see, e.g., [13].

To determine the basis size we try to satisfy

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

with $p_e \geq 2$ to ensure that μ_{p+1} converges (since it can be a complex or multiple eigenvalue). In fact, larger values of p_e greatly improve the convergence speed of the subspace iteration procedure. In addition, to improve robustness in the calculation of the Floquet multipliers outside C_ρ, components of $V_e^{[0]}$ are randomly perturbed at the start of each new continuation step. This proved to be reliable in our test cases. It avoids the problem encountered in the second test example of [20], where a Hopf bifurcation point is detected too late along a continuation branch. It is also much simpler than criteria based on monitoring the convergence speed as in [8, 20], since slow convergence can also be caused by an inaccurate basis or bad starting values.

To conclude this section, we briefly sketch the final algorithm for NPGS(2) using subspace iteration with projection and discuss the work per iteration step.

ALGORITHM 3.2. *NPGS(2) and subspace iteration with projection for computing a periodic solution (without locking).*

Input: *Starting value* $x^{(0)}(0)$, $T^{(0)}$.
Starting basis $V_e = \begin{bmatrix} v_1 & \cdots & v_{p+p_e} \end{bmatrix}$
routine to compute $M(x(0),T)v$.

Output: *FAILURE or* $x^*(0)$, T^*, *final basis* V_e.

begin
  $\nu = 0$
  repeat
    Compute $\varphi \leftarrow \varphi(x(0),T)$; $r \leftarrow \varphi - x(0)$
    for $i = 1$ **to** ν_{sub} **do**
      $W_e \leftarrow M(x(0),T)V_e$
      $S_e \leftarrow V_e^T W_e$
      Compute the ordered Schur decomposition $S_e Y_e = Y_e R_e$
      if *(* $i < \nu_{sub}$ *)* **then**
        $V_e \leftarrow W_e Y_e$
        Orthonormalize V_e
      end if
    end for
    Determine the value of p.

$V_p \leftarrow V_e Y_e[1 \cdots p]$
Compute the Picard correction:

$$\begin{aligned} \Delta q &\leftarrow \left(I - V_p V_p^T\right) r \\ \Delta q &\leftarrow \left(I - V_p V_p^T\right) \left(M(x(0), T)\Delta q + r\right). \end{aligned}$$

Compute the Newton correction:

$$\begin{bmatrix} R_e[1 \cdots p,\, 1 \cdots p] - I_p & V_p^T f(\varphi) \\ c_1^T(x(0), T) V_p & d_{1\,1}(x(0), T) \end{bmatrix} \begin{bmatrix} \Delta \bar{p} \\ \Delta T \end{bmatrix} = - \begin{bmatrix} V_p^T \left(r + M(x(0), T)\Delta q\right) \\ s(x(0), T) + c_1^T(x(0), T)\Delta q \end{bmatrix}.$$

Update the point:

$$\begin{aligned} x(0) &\leftarrow x(0) + \Delta q + V_p \Delta \bar{p} \\ T &\leftarrow T + \Delta T. \end{aligned}$$

$V_e = W_e Y_e$
Decide whether vectors should be added to or deleted from V_e.
Orthonormalize V_e
$\nu \leftarrow \nu + 1$
until $\nu \geq \nu_{max}$ **or** *convergence*
if *(no convergence)* **then return** *FAILURE*
else $x^*(0) \leftarrow x(0)$; $T^* \leftarrow T$;
end

The algorithm as given above requires $\nu_{sub}(p + p_e) + 3$ IVP solves per iteration: one to compute $\varphi(x(0), T)$, one additional IVP solve to compute the Picard correction, one to compute the residual for the Newton part, and $\nu_{sub}(p + p_e)$ to build the basis. In our tests, we used $\nu_{sub} = 1$, except for the first iteration step. The time integration needed in the NPGS(l) schemes to compute the right-hand side of the Newton part is not needed in the NPJ(l)-schemes. The CNP(l) scheme needs $\nu_{sub}(p + p_e) + 2l + 1$ IVP solves per iteration since the Picard scheme (2.12) must be executed twice per iteration step. When using our methods to compute branches of periodic solutions, we advise the use of the NPGS(l) scheme wherever possible and a switch to the CNP(l) scheme near a turning point. A turning point might occur when there is a double Floquet multiplier at $+1$. One can develop a detection criterion based on monitoring the available eigenvalues and switching to CNP(l) when an eigenvalue (except for the trivial Floquet multiplier) approaches 1. This means, however, that we switch to the CNP(l) scheme also in the neighborhood of transcritical or pitchfork bifurcations.

At this point it is worth emphasizing again that our overall strategy is tuned to utilize good starting values available because of the continuation context, and the algorithm is a balance between efficiency and reliability. In particular, if a failure occurs then the increment in the continuation parameter is halved and the computations restarted.

4. Numerical results. In this section we illustrate the theoretical material presented in this paper by applying our algorithms to two specific example systems of PDEs. The systems, presented in section 4.1 below, are derived from models of chemical kinetics and visco-elastic fluid dynamics, respectively. For each model we compare the efficiency of each of the algorithms presented in this paper in reproducing known bifurcation diagrams of periodic orbits.

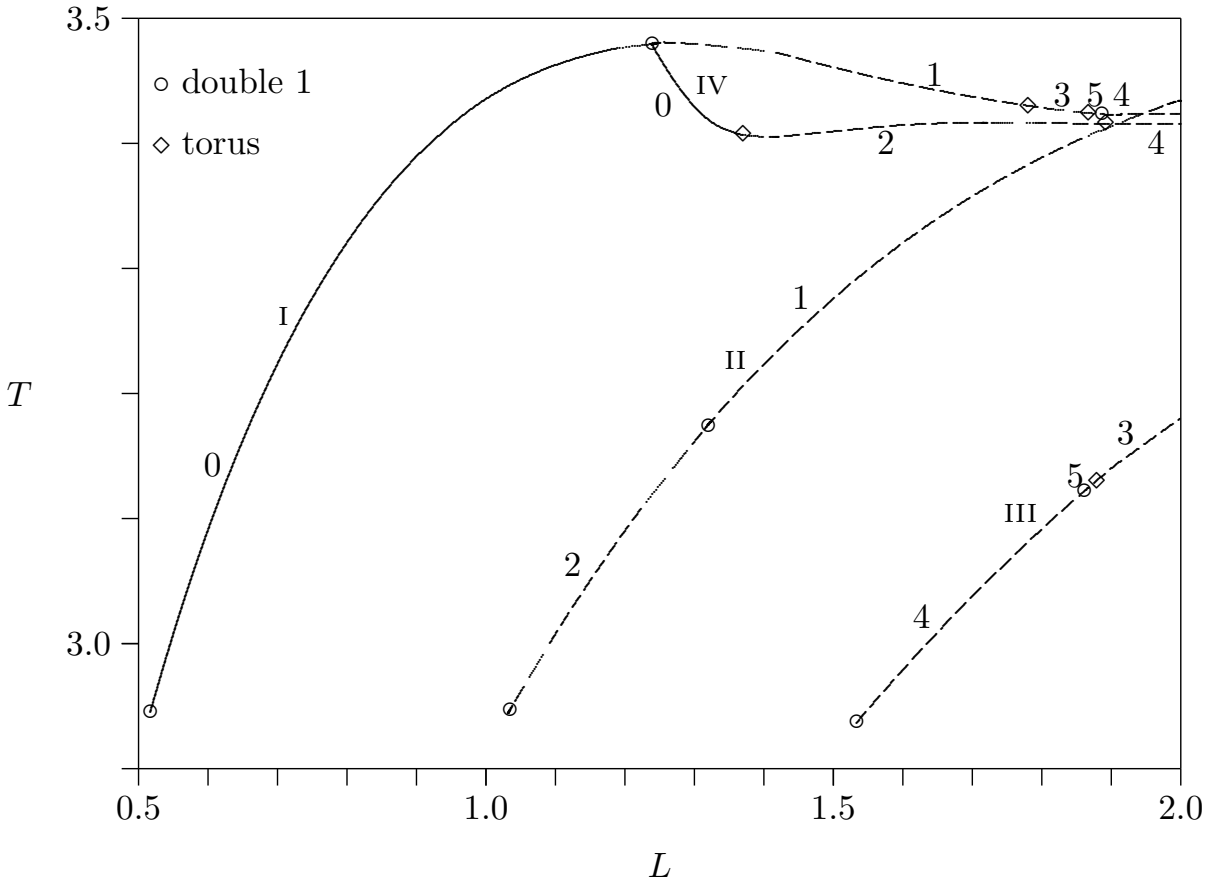

FIG. 4.1. *Periodic solutions bifurcation diagram for the discretized Brusselator model* ($h = \frac{1}{32}$)*, period* T *versus the reactor length* L*. Arabic numbers indicate the number of Floquet multipliers outside the unit circle. Roman numbers indicate the number of the branch used in this section's tables. Double* 1 *Floquet multipliers and torus bifurcations are marked with* ○ *and* ◇*, respectively. No period doublings occur on the computed branches. Note that we did not compute the branches intersecting branch* I *around* $L = 1.89$*, branch* II *around* $L = 1.32$*, and branch* III *around* $L = 1.86$*.*

4.1. Models. The first model is the one-dimensional *Brusselator* [5]:

$$\begin{aligned} \frac{\partial X}{\partial t} &= \frac{D_X}{L^2}\frac{\partial^2 X}{\partial z^2} + X^2Y - (B+1)X + A, \\ \frac{\partial Y}{\partial t} &= \frac{D_Y}{L^2}\frac{\partial^2 Y}{\partial z^2} - X^2Y + BX, \end{aligned} \tag{4.1}$$

with Dirichlet boundary conditions

$$\begin{aligned} X(t, z=0) = X(t, z=1) = A, \\ Y(t, z=0) = Y(t, z=1) = \frac{B}{A}. \end{aligned} \tag{4.2}$$

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 $D_Y = 0.004$. Branches of periodic solutions bifurcate from the trivial steady-state branch ($X = A, Y = \frac{B}{A}$), at Hopf bifurcation points at $L_k^H = k\,0.5130$, $k = 1, 2, \ldots$. For the reported results, we used an $O(h^2)$ finite difference space discretization with grid size $h = \frac{1}{32}$, yielding a system of ordinary differential equations (ODEs) of dimension $N = 62$. Four branches of the bifurcation diagram obtained with this discretization are given in Figure 4.1.

The second test problem is the *Olmstead model* for fluid flow with memory, given in [14]. It can be written as a system of two PDEs

$$\begin{aligned} \frac{\partial u}{\partial t} &= (1-\delta)\frac{\partial^2 v}{\partial x^2} + \delta\frac{\partial^2 u}{\partial x^2} + Ru - u^3, \\ \lambda\frac{\partial v}{\partial t} &= u - v, \end{aligned} \tag{4.3}$$

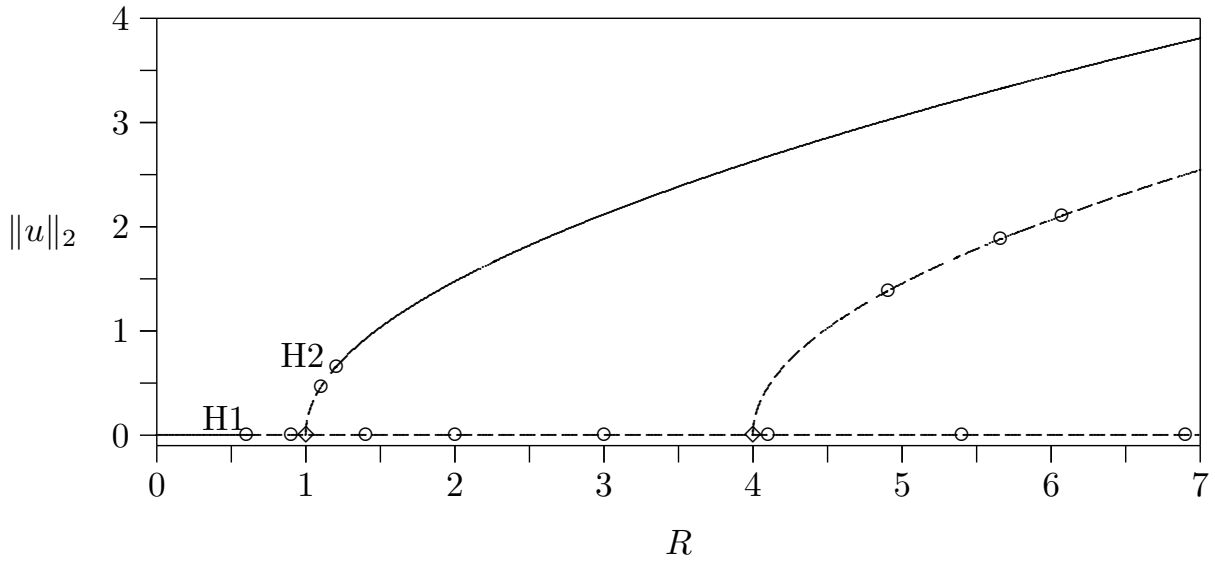

FIG. 4.2. *Steady-state bifurcation diagram for the Olmstead model, two-norm of the solution profile versus the parameter R. Hopf bifurcation points are marked with ∘, pitchfork bifurcations with ⋄.*

with boundary conditions

$$\begin{aligned} u(t, x = 0) &= u(t, x = \pi) = 0, \\ v(t, x = 0) &= v(t, x = \pi) = 0. \end{aligned} \tag{4.4}$$

The Rayleigh number R is used as the bifurcation parameter. The values of the parameters λ (a relaxation time) and δ (the ratio of the retardation time to the relaxation time) are kept fixed at $\lambda = 2.0$ and $\delta = 0.1$. The steady-state bifurcation diagram is shown in Figure 4.2. As shown in [14], the system has a trivial steady-state solution branch at $u = v = 0$ with pitchfork bifurcations at $R = k^2$, $k = 1, 2, \ldots$ and Hopf bifurcations at

$$R = \frac{1}{\lambda} + k^2\delta = 0.5 + k^2 0.1, \qquad k = 1, 2, \ldots .$$

On the steady-state solution branch bifurcating from the trivial branch at $R = 1$, Hopf points arise at $R \approx 1.1007$ and $R \approx 1.2040$. We computed parts of the (initially stable) periodic solution branch, bifurcating from the trivial branch at $R = 0.6$, and of the (unstable) branch emanating at $R = 1.2040$ on the first bifurcated steady-state branch. We will label these branches in the tables with "branch 1" and "branch 2," respectively. For the space discretization we used an $O(h^2)$ finite difference discretization resulting in a system of 80 ODEs.

4.2. Robustness under discretization. In general, when the space discretization of a PDE is refined, additional eigenvalues appear at the left of the spectrum of the Jacobian (corresponding to rapidly decaying modes) while the eigenvalues on the right-hand side of the spectrum shift only slightly (assuming the coarse discretization has captured the essential features of the physical problem sufficiently well). For periodic solutions, this phenomenon translates into the addition of Floquet multipliers close to zero. The dominant Floquet multipliers remain almost unchanged. We illustrate this phenomenon in Figure 4.3. It shows the Floquet multipliers for a solution on the first branch of the Brusselator (4.1) with $L \approx 1.49$, discretized with two different grid sizes. As a result, the basis size needed in our algorithms to achieve a given convergence speed, and the convergence speed of subspace iteration are virtually independent of N. Since the cost of the linear algebra operations in our methods grows

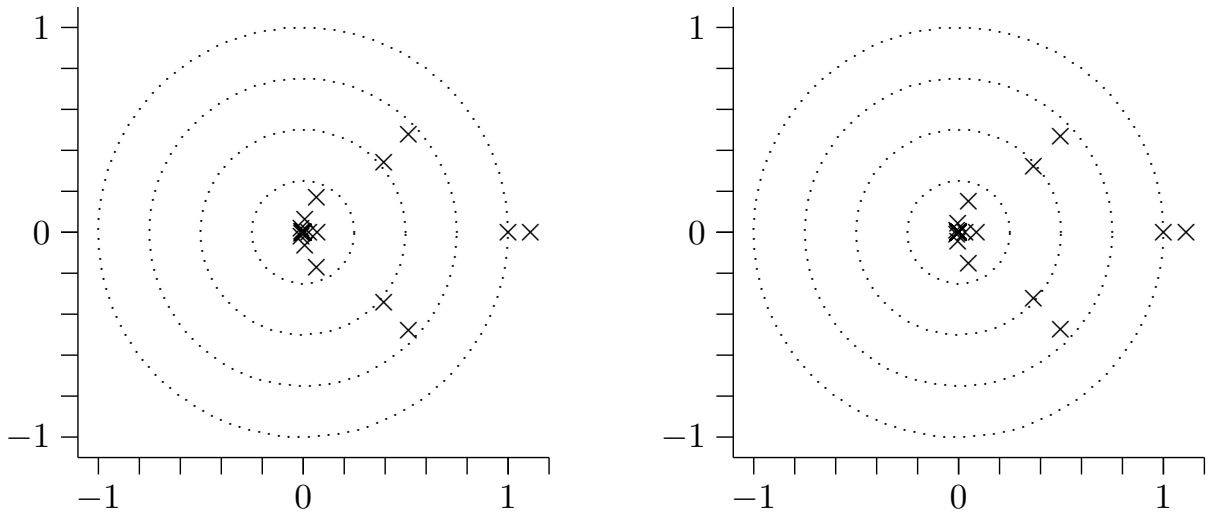

FIG. 4.3. *Floquet multipliers of a periodic solution of the Brusselator, first branch (emanating at $L = 0.55$), 15 (left) and 31 (right) discretization points, $L = 1.4905$ (left) and $L = 1.4833$ (right).*

only linearly with N, the computational complexity of our algorithms is completely determined by that of the IVP solver. For shooting methods using a traditional Newton–Raphson solver—in which the full Jacobian matrix must be computed—the computational complexity is N times higher (or even more if the cost of the linear system solver would become dominant, which is very unlikely). We can conclude that in our Newton–Picard methods the number of IVP solves is fully determined by the system's dynamics, while in Newton's method the number of degrees of freedom of the discretization determines the number of IVP solves. Hence Newton–Picard methods are likely to be more competitive when a high accuracy and thus a fine discretization is required.

4.3. Discussion of results. The influence of the threshold ρ on the numerical performance was discussed in [17] for the Brusselator example. The conclusion there was that values of ρ around 0.5 give good performance. A larger ρ results in a smaller basis but slower convergence speed. The value of ρ is somewhat problem dependent but, experience has shown, is in fact not very critical.

For our current tests, we have used $\rho = 0.5$ and 4 extra vectors (i.e., $p_e = 4$; see section 3.2). Using four extra vectors instead of two, as done in [17], greatly improves the convergence and the robustness of the subspace iterations and results in fewer IVP solves overall. The use of further extra vectors or smaller values of ρ did not result in further improvements in our test examples. This can easily be explained heuristically using Figure 4.3; by using $\rho = 0.5$ and four extra vectors, we generally capture all basis vectors outside the cluster around 0. The convergence factor becomes so small that the asymptotic rate is not reached at all. In fact the Q-projection of the residual decreases faster than the P-projection in the initial iterations, which means that the behavior of the Newton component becomes dominant.

We used a linear phase condition based on the starting value $(x(0)^{(0)}, \lambda^{(0)})$ for the Newton–Picard iterations:

$$f(x(0)^{(0)}, \lambda^{(0)})^T (x(0) - x(0)^{(0)}) = 0. \tag{4.5}$$

This condition defines a $(N-1)$-dimensional hyperplane (called the Poincaré return map) and should intersect the limit cycle transversally, which requires

$$f(x(0)^{(0)}, \lambda^{(0)})^T f(x(0)^*, \lambda^*) \neq 0$$

(see, e.g., [3, 19]). Equation (4.5) satisfies this requirement for reasonable starting values.

We computed several branches of periodic solutions, using the various algorithms within a simple variable stepsize continuation code based on the strategy used in Locbif [12]. A maximum and minimum predictor stepsize is user imposed. After a convergence failure, the stepsize is halved. The computations are stopped when the minimum stepsize is reached. When a point is computed sufficiently fast, the stepsize is increased by a factor depending on the convergence history for the previous points and the maximum stepsize. We do not use an angular control criterion as in Locbif since such criteria performed badly in experiments with large-scale systems. The variable stepsize strategy allows us to observe not only differences in (asymptotic) convergence speed but also differences in the size of the attraction domain of the different methods. Some of the reported results use pseudo-arclength parameterization. Two linear parameterization equations were tested:

$$n_1\left(x(0),T,\lambda\right) = \left(x(0)-x_p(0)\right)^T x_s(0) + \Theta_T(T-T_p)T_s + \Theta_\lambda(\lambda-\lambda_p)\lambda_s \tag{4.6}$$

and

$$\begin{aligned} n_2\left(x(0),T,\lambda\right) &= \left(x(0)-x_p(0)\right)^T \frac{x_s^T(0)\,f(x_p(0))}{f^T(x_p(0))\,f(x_p(0))} f(x_p(0)) \\ &\quad +\Theta_T(T-T_p)T_s + \Theta_\lambda(\lambda-\lambda_p)\lambda_s. \end{aligned} \tag{4.7}$$

$(x_s(0),T_s,\lambda_s)$ is the predictor step along the secant vector and $(x_p(0),T_p,\lambda_p)$ the predicted point. Θ_T and Θ_λ are two positive scaling parameters. Condition (4.7) is derived from (4.6) by projecting the $x(0)$-components of the secant vector on the tangent vector of the trajectory starting in the predicted point.

The computations were done on several variants of the method, using different values for the parameters. Here we summarize the main conclusions. The main cost in the overall scheme is the number of IVP solves, and in Table 4.1 we use this measure to compare the following five methods.

- NPJ (Newton–Picard Jacobi), based on (2.12) and

$$\begin{bmatrix} V_p^T M V_p - I_p & V_p^T b_1 \\ c_1^T V_p & d_{1\,1} \end{bmatrix} \begin{bmatrix} \Delta\bar{p} \\ \Delta T \end{bmatrix} = - \begin{bmatrix} V_p^T r \\ s \end{bmatrix};$$

- NPGS (Newton–Picard Gauss–Seidel), based on (2.12) and (2.13).
- CNPGS, namely continuation Newton–Picard Gauss–Seidel. This method uses the Jacobian approximation

$$\begin{bmatrix} -\left(\sum_{i=0}^{l-1}(V_q^T M V_q)^i\right)^{-1} & 0 & 0 & V_q^T b_2 \\ 0 & V_p^T M V_p - I_p & V_p^T b_1 & V_p^T b_2 \\ 0 & c_1^T V_p & d_{1\,1} & d_{1\,2} \\ 0 & c_2^T V_p & d_{2\,1} & d_{2\,2} \end{bmatrix}.$$

Here we first solve the equations for $\Delta\bar{p}$, ΔT, and $\Delta\lambda$ and then compute $\Delta\bar{q}$ from the Picard scheme

$$\begin{cases} \Delta\bar{q}^{[0]} = 0, \\ \Delta\bar{q}^{[i]} = V_q^T M V_q \Delta\bar{q}^{[i-1]} + V_q^T(r + b_2\Delta\lambda), \quad i = 1,\ldots,l, \\ \Delta\bar{q} = \Delta\bar{q}^{[l]} = \sum_{i=0}^{l-1}(V_q^T M V_q)^i V_q^T(r + b_2\Delta\lambda). \end{cases}$$

TABLE 4.1
Number of IVP solves (best results obtained over 12 continuation runs).

Method	Brusselator model				Olmstead model		Total
Branch	I	II	III	IV	1	2	
Start at $L/R =$	5.55	1.04	1.55	1.30	0.623	1.209	
$T =$	3.017	2.948	2.947	3.433	14.06	14.27	
End at $L/R =$	2.0	2.0	2.0	2.0	1.267	1.24	
$T =$	3.424	3.436	3.197	3.415	27.5	18.0	
NPJ	1045	1611	959	1463	864	385	6327
NPGS	723	721	682	950	832	412	4320
CNPGS	905	852	746	919	1238	539	5199
CNP	987	932	753	998	1408	662	5740
Average	915	1029	785	1083	1086	500	5397
Chord–Newton	2053	1607	1128	1874	9229	3791	19682

Since

$$r + b_2 \Delta\lambda = r(x(0), T, \lambda + \Delta\lambda) + \text{h.o.t.},$$

this is again essentially a Gauss–Seidel approach where we first solve the P-system for Δp, ΔT, and $\Delta\lambda$ and then use the update λ to compute Δq. CNPGS(1) applied to the equilibrium point problem (2.4) is the method presented in [20, section 7]. We used the pseudo-arclength condition (4.7).

- CNP, based on the Sherman–Morrison formula explained in (2.15)–(2.18) and (4.7).

The fifth method is a direct application of a Chord–Newton method on (1.4) which does not use the splitting approach. This is included for comparison purposes. To be more precise,

- Chord–Newton applied to (1.4) directly. We tried several strategies for updating the Jacobian matrix and only report the best result in Table 4.1. Here we used the parameterization equation (4.6).

All Newton–Picard methods are implemented similarly to Algorithm 3.2, but we used subspace iteration with locking. We always used all vectors from V_e in V_p. We did one subspace iteration step before each Newton–Picard step, except before the first Newton–Picard iteration step for the first point on a branch, where we used a criterion on the accuracy of the computed subspace. All matrix–vector products and the Jacobian matrices for the Chord–Newton method were computed using variational equations.

Table 4.1 lists the number of IVP solves for each method on the two problems. To be precise, we computed the branches using 1, 2, or 3 Picard steps and 4 values for the maximal steplength of the continuation code but report only the best result of the 12 combinations in the table. The last column lists the total number of IVP solves. It is seen that NPGS performs best overall and NPJ worst, being about 50% more expensive than NPGS. Note, however, that the difference in performance occurs for the Brusselator example and that for the Olmstead model the performance of both methods is the same. This superiority of NPGS is mainly due to the larger domain of attraction and higher convergence performance far from the solution. Close to a solution, the difference in convergence speed between the Gauss–Seidel and Jacobi variant was found to be negligible, in agreement with Corollary 2.4. The Gauss–Seidel variant thus allows larger stepsizes (and hence fewer continuation points) in a continuation procedure. As would be expected CNP, which uses the rank-one update

TABLE 4.2
Number of continuation points (best results obtained over 12 continuation runs).

Method	Brusselator model				Olmstead model		Total
Branch	I	II	III	IV	1	2	
NPJ	16	21	11	18	16	7	89
NPGS	11	9	7	12	15	7	61
CNPGS	13	9	7	11	23	8	71
CNP	13	9	7	11	23	9	72
Average	13	12	8	13	19	8	73
Chord–Newton	9	7	5	8	22	9	60

TABLE 4.3
Value for l used to obtain the results presented in the previous tables.

Method	Brusselator model				Olmstead model		Average
Branch	I	II	III	IV	1	2	
NPJ	2	1	1	1	1	1	1.17
NPGS	2	2	3	1	1	1	1.67
CNPGS	1	1	3	2	1	1	1.50
CNP	1	1	2	1	1	1	1.17

approach, and CNPGS are both more expensive than NPGS. Compared with the CNPGS method, the additional IVP solves per iteration needed in the CNP method because of the full coupling do not result in improved convergence speed. However, the CNP method proves to be the most robust method. As expected Chord–Newton is the most expensive method, but it is significant that even for these relatively small problems NPGS is over four times faster than Chord–Newton. For larger N the superiority would be even more marked because of the scaling discussion in section 4.2. Table 4.2 compares the same methods with the number of continuation steps taken to compute the branch. Similar conclusions can be drawn from this table.

For the four Newton–Picard methods, the number of Picard steps to compute $\Delta\bar{q}$ in (2.12) is varied from $l = 1$ to $l = 3$. Table 4.3 indicates which value for l provided the fastest performance, the figures for which were used to produce Tables 4.1 and 4.2. For the Olmstead model it is striking that $l = 1$ was best for all runs. For the Brusselator, the results are less clear cut, though the choice $l = 1$ was best in 9 out of 16 runs. There are two possible explanations for this. First, for our choice of parameters the theoretical convergence rate is quite high and is not reached in practice even for $l = 1$. In fact, the transitional behavior of the Newton part often dominates the convergence. Second, the convergence speed of the subspace iteration procedure does not change when l is changed. The basis is not accurate enough to get the full theoretical convergence speed.

The number of IVP solves presented in Table 4.1 includes the accurate computation of all Floquet multipliers larger than 0.7. After the Newton–Picard iterations (where the basis update is stopped once two to three digits of accuracy for the eigenvalues is reached) we generally needed only one or two subspace iteration steps to compute the dominant Floquet multipliers with four digits of accuracy. Note that for the Olmstead model, we have observed inaccuracies in the computation of the trivial one eigenvalue near Hopf bifurcations or for solutions with a large period. The computation of the Floquet multipliers by applying the QR algorithm (from LAPACK) to the full monodromy matrix resulted in comparable or even larger errors. This is

clearly caused by inaccuracies in the computation of the Jacobian. Compared with the subspace method, the use of the QR algorithm on the full monodromy matrix also results in larger differences in accuracy between approaches using finite differences and using the variational equations.

5. Conclusions. In this paper we have developed Newton–Picard methods for the computation of periodic solutions of PDEs or large-scale systems of ODEs, based on single shooting. These methods are extensions of the recursive projection method described in [20]. Our approach utilizes the fact that, in continuation, nearby starting values are likely to be available, and if all else fails one can reduce the steplength in the continuation parameter and restart the process. These methods are particularly efficient for systems which exhibit only low-dimensional dynamics (as is the case for many physically interesting systems) where the number of large Floquet multipliers is essentially independent of the discretization, and this is exploited in our methods. We have also shown how the Newton–Picard schemes could be implemented with (pseudo-arclength) continuation.

We have tested the various Newton–Picard schemes for the computation of branches of periodic solutions for two model problems. The test results indicate that the Gauss–Seidel variants (e.g., the NPGS(1) scheme) are superior to the Jacobi variants (e.g., NPJ(1)) . Although the asymptotic convergence speed is the same, the Gauss–Seidel variants have in general a larger domain of attraction. Compared with classical shooting and (Chord–) Newton methods, which need the explicit computation and factorization of the whole Jacobian matrix, Newton–Picard schemes are in general much more efficient. Newton–Picard schemes also provide good approximations for the dominant Floquet multipliers, which can be refined by a few subspace iteration steps. These Floquet multipliers are used to determine stability and could easily be used to accurately detect bifurcations.

As is the case with all variants of the simple shooting idea, the Newton–Picard schemes presented here for the computation of periodic solutions have the advantage that memory requirements remain low, but the disadvantage is that it is hard or impossible to compute unstable limit cycles with large Floquet multipliers or limit cycles with long period. We believe that the ideas behind our methods can also be applied to multiple shooting algorithms [11, 19] to overcome these disadvantages.

REFERENCES

[1] V. I. Arnold, *Ordinary Differential Equations*, MIT Press, Cambridge MA, 1973.

[2] E. Doedel, *AUTO: Software for Continuation and Bifurcation Problems in Ordinary Differential Equations*, Tech. report, Applied Mathematics, California Institute of Technology, Pasadena, 1986.

[3] T. Fairgrieve, *The Computation and Use of Floquet Multipliers for Bifurcation Analysis*, Ph.D. thesis, Department of Computer Science, University of Toronto, 1994.

[4] T. Fairgrieve and A. Jepson, *O.K. Floquet multipliers*, SIAM J. Numer. Anal., 28 (1991), pp. 1446–1462.

[5] M. Holodniok, P. Knedlik, and M. Kubicek, *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.

[6] H. Jarausch, *Zur numerischen Untersuchung von parabolischen Differentialgleichungen mit Hilfe einer adaptiven spektralen Zerlegung*, Habilitationsschrift, RWTH Aachen, January 1991.

[7] H. Jarausch, *Analyzing stationary and periodic solutions of systems of parabolic partial differential equations by using singular subspaces as reduced basis*, Bericht Nr. 92, Institut für Geometrie und Praktische Mathematik, RWTH Aachen, 1993.

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

[9] H. Jarausch and W. Mackens, *Computing bifurcation diagrams for large nonlinear variational problems*, in Large Scale Scientific Computing, P. Deuflhard and B. Enguist, eds., Progress in Scientific Computing 7, Birkhäuser-Verlag, Basel, 1987.

[10] H. Jarausch and W. Mackens, *Solving large nonlinear systems of equations by an adaptive condensation process*, Numer. Math., 50 (1987), pp. 633–653.

[11] H. Keller, *Numerical Methods for Two-Point Boundary Value Problems*, Blaisdell, New York, 1968.

[12] A. Khibnik, Y. Kuznetsov, V. Levitin, and E. Nikolaev, *Continuation techniques and interactive software for bifurcation analysis of ODEs and iterated maps*, Physica D, 62 (1993), pp. 360–371.

[13] K. Meerbergen and D. Roose, *Matrix transformations for computing rightmost eigenvalues of large sparse nonsymmetric eigenvalue problems*, IMA J. Numer. Anal., 16 (1996), pp. 297–346.

[14] W. Olmstead, S. Davis, S. Rosenblat, and W. Kath, *Bifurcation with memory*, SIAM J. Appl. Math., 4 (1986), pp. 171–188.

[15] W. Rheinboldt, *Numerical analysis of continuation methods for nonlinear structural problems*, Comput. & Structures, 13 (1981), pp. 103–113.

[16] W. Rheinboldt, *Numerical Analysis of Parameterised Nonlinear Equations*, Wiley-Interscience, New York, 1986.

[17] 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.

[18] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Algorithms and Architectures for Advanced Scientific Computing, Manchester University Press, Manchester, 1992.

[19] R. Seydel, *Practical Bifurcation and Stability Analysis. From Equilibrium to Chaos*, 2nd ed., Springer-Verlag, New York, 1994.

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

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